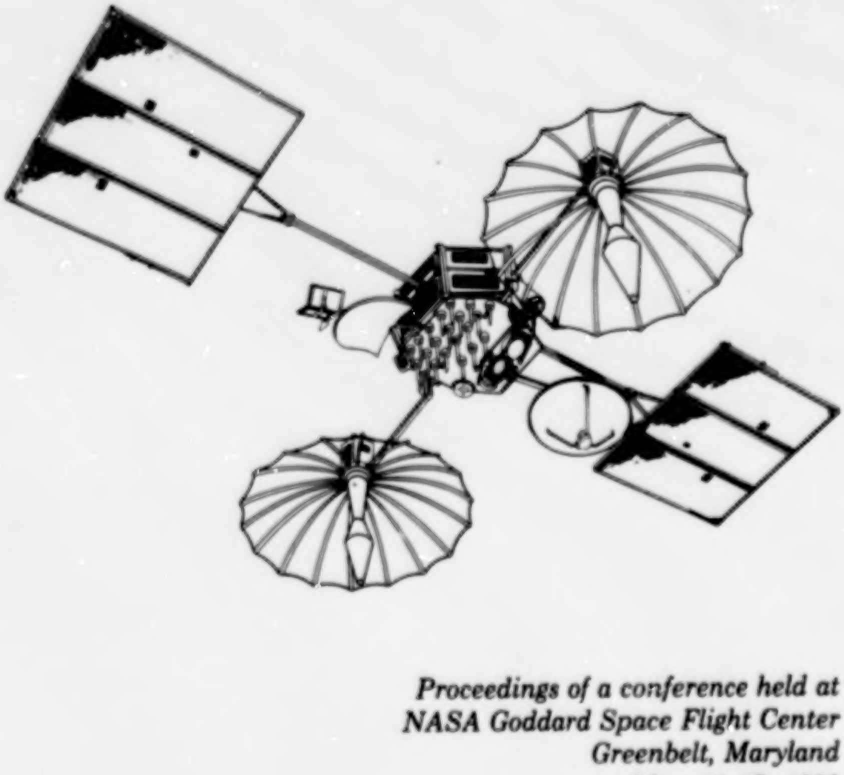
MF-0830-H-10
AUG 19 1989 (327)
NAS 1.55:3033

NASA Conference Publication 3033

1989 Goddard Conference on Space Applications of Artificial Intelligence



Proceedings of a conference held at
NASA Goddard Space Flight Center
Greenbelt, Maryland
May 16-17, 1989

NASA Conference Publication 3033

1989 Goddard Conference on Space Applications of Artificial Intelligence

Edited by
James Rash
Goddard Space Flight Center
Greenbelt, Maryland

Proceedings of a conference sponsored by
NASA Goddard Space Flight Center, Mission
Operations and Data Systems Directorate,
Greenbelt, Maryland, and held at
NASA Goddard Space Flight Center
Greenbelt, Maryland
May 16–17, 1989

NASA
National Aeronautics and
Space Administration
Office of Management
Scientific and Technical
Information Division
1989

Conference Committee

Carolyn Dent (Chair), GSFC
Troy Ames, GSFC
Lisa Basile, GSFC
David Beyer, Bendix Field Engineering Corp.
Michael Bracken, GSFC
Joy Bush, CSC
William Campbell, GSFC
Elizabeth Chandler, GSFC
Robert Cromp, Science Applications Research, Inc.
Robert Dutilly, GSFC
Peter Hughes, GSFC
Larry Hull, GSFC
James Rash, GSFC
Walter Truszkowski, GSFC

Foreword

Welcome to the 1989 Goddard Conference on Space Applications of Artificial Intelligence (AI). This is Goddard's fourth AI Conference and we are still experimenting with our presentation format. The first day of the conference, May 16, consists of poster session presentations to allow maximum interaction between presenters and attendees. On the second day, May 17, special interest session presentations are scheduled, each concentrating on a specific artificial intelligence application area: Near Real Time Computing; Planning and Scheduling; Monitoring and Fault Diagnosis; and Intelligent User Interfaces. Each session includes several papers representing different viewpoints in the application area, and offers opportunities for the audience to participate in discussion and debate of these different viewpoints.

One of the things which is unique about our conference is the focus on space applications of artificial intelligence. All the papers in these proceedings, even the more theoretical papers, maintain this focus. Among the papers in the proceedings you may find a few familiar applications such as Fault Isolation Expert System for TDRSS Applications (FIESTA) and the Plan Specification Tool (PST) - Planning and Resource Reasoning (PARR), formerly known as Interactive Experimenter Planning System (IEPS). The applications have been enhanced and have matured to operational expert systems. You will also find new applications such as Request Oriented Scheduling Engine (ROSE) and the Knowledge-Based Geographic Information System (KBGIS). These applications are currently prototype systems and we look forward to following their development in future AI conferences.

I would like to thank all the other members of the 1989 Goddard AI Conference Planning Committee whose names are elsewhere listed. Special thanks to the following companies for their support: Bendix Field Engineering Corporation, Computer Sciences Corporation, Martin Marietta Data Systems, and Science Applications Research, Incorporated. And finally thanks to all the presenters and attendees for contributing to the success of our AI Conference.

Carolyn P. Dent
Conference Planning Committee Chair

Preface

The 1989 Goddard Conference on Space Applications of Artificial Intelligence, May 16 and 17, 1989 at the Goddard Space Flight Center, Greenbelt, Maryland, has international participation from industry, academia, and NASA.

With continuing sponsorship by the Mission Operations and Data Systems Directorate at Goddard, the primary purpose of the conference remains to facilitate the communication of results among groups working on space applications of artificial intelligence. An additional purpose is to provide a forum for recognition of contributors in this field. One measure of the success of this conference will be the degree to which it stimulates contributions to our future conferences.

The emphasis again this year is on space applications; but theoretical work is also represented because advances in the theoretical and applied domains are necessarily connected.

Planning for the 1990 Goddard AI Conference is under way. Contributions in the form of technical papers are again broadly encouraged from groups and individuals concerned with space applications of artificial intelligence.

Table of Contents

Mission Operations Support

Knowledge Based and Interactive Control for the Superfluid Helium On-Orbit Transfer Project 3
Timothy P. Castellano, Eric A. Raymond, Jeff C. Shapiro, Frank A. Robinson, Donald A. Rosenthal

Shared Resource Control Between Human and Computer 13
James Hendler, Reid Wilson

An English Language Interface for Constrained Domains 21
Brenda J. Page

Planning and Scheduling

Ground Data Systems Resource Allocation Process 37
Carol A. Berner, Ralph Durham, Norman B. Reilly

A Situated Reasoning Architecture for Space-based Repair and Replace Tasks 49
Ben Bloom, Debra McGrath, Jim Sanborn

Parallel Plan Execution with Self-Processing Networks 61
C. Lynne D'Autrechy, James A. Reggia

Mission Scheduling 75
Christine Gaspin

On the Development of a Reactive Sensor-Based Robotic System 87
Henry H. Hexmoor, William E. Underwood, Jr.

PST & PARR: Plan Specification Tools and A Planning and Resource Reasoning Shell for Use in Satellite Mission Planning 101
David McLean, Wen Yen

An Approach to Knowledge Engineering to Support Knowledge-Based Simulation of Payload Ground Processing at the Kennedy Space Center 113
Shawn McManus, Michael McDaniel

A Heuristic Approach to Incremental and Reactive Scheduling 127
Jidé B. Odubiyi, David R. Zoch

Fault Isolation/Diagnosis

A Method for Interactive Satellite Failure Diagnosis: Towards a Connectionist Solution 143
P. Bourret, J. A, Reggia

BCAUS Project Description and Consideration of Separation of Data and Control 153
Joy L. Bush, Steven J. Weaver

Tracking & Data Relay Satellite Fault Isolation & Correction Using PACES: Power & Attitude Control Expert System 161
Carol-Lee Erikson, Peggy Hooker

Spacelab Life Sciences-1 Electrical Diagnostic Expert System 169
C. Y. Kao, W. S. Morris

SHARP: A Multi-Mission AI System for Spacecraft Telemetry Monitoring and Diagnosis 185
Denise L. Lawson, Mark L. James

REDEX: The Ranging Equipment Diagnostic Expert System 201
Edward C. Luczak, K. Gopalakrishnan, David J. Zillig

Diagnostic Tolerance for Missing Sensor Data 213
Ethan A. Scarl

Image Processing and Machine Vision

Robot Acting on Moving Bodies (RAMBO): Preliminary Results 223
Larry S. Davis, Daniel DeMenthon, Thor Bestul, Sotirios Ziavras, H. V. Srinivasan, Madhu Siddalingaiah, David Harwood

Data Management

Development of an Intelligent Interface for Adding Spatial Objects to a Knowledge-Based Geographic Information System 239
William J. Campbell, Craig Goettsche

The Utilization of Neural Nets in Populating an Object-Oriented Database 249
William J. Campbell, Scott E. Hill, Robert F. Cromp

A Rapid Prototyping/Artificial Intelligence Approach to Space Station-Era Information Management and Access 265
Richard S. Carnahan, Jr., Stephen M. Corey, John B. Snow

An Intelligent User Interface for Browsing Satellite Data Catalogs 281
Robert F. Cromp, Sharon Crook

Natural Language Processing and Advanced Information Management 301
James E. Hoard

Modeling and Simulation

A Logical Model of Cooperating Rule-Based Systems 319
Sidney C. Bailin, John M. Moore, Robert H. Hilberg, Elizabeth D. Murphy, Shari-Ann Bahder

Synthetic Organisms and Self-Designing Systems 335
W. B. Dress

Integration of Perception and Reasoning in Fast Neural Modules 349
David G. Fritz

A Connectionist Model for Dynamic Control 357
Kevin C. Whitfield, Sharon M. Goodall, James A. Reggia

Development Tools/Methodologies

Expert System Development Methodology and the Transition from Prototyping to Operations: FIESTA, A Case Study 373
Nadine Happell, Steve Miksell, Candace Carlisle

Applications of Fuzzy Sets to Rule-Based Expert System Development 385
Robert N. Lea

Late Submission

Genetic Algorithms Applied to the Scheduling of the Hubble Space Telescope 391
Jeffrey L. Sponsler

Mission Operations Support

BLANK PAGE

KNOWLEDGE BASED AND INTERACTIVE CONTROL FOR THE SUPERFLUID HELIUM ON-ORBIT TRANSFER PROJECT

Timothy P. Castellano,[§] Eric A. Raymond,[§] Jeff C. Shapiro[§]
Frank A. Robinson,[§] and Donald A. Rosenthal

Artificial Intelligence Research Branch
NASA Ames Research Center
Mail Stop 244-17
Moffett Field, CA 94035

timothy@pluto.arc.nasa.gov
(415) 694-3180

Abstract

NASA's Superfluid Helium On-Orbit Transfer (SHOOT) project is a Shuttle-based experiment designed to acquire data on the properties of superfluid helium in micro-gravity. Aft Flight Deck Computer Software for the SHOOT experiment is comprised of several monitoring programs which give the astronaut crew visibility into SHOOT systems and a rule based system which will provide process control, diagnosis and error recovery for a helium transfer without ground intervention. Given present Shuttle manifests, this software will become the first expert system to be used in space. The SHOOT Command and Monitoring System (CMS) software will provide a near real time highly interactive interface for the SHOOT principal investigator to control the experiment and to analyze and display its telemetry. The CMS software is targeted for all phases of the SHOOT project: hardware development, pre-flight pad servicing, in-flight operations, and post-flight data analysis.

Introduction

The SHOOT experiment is a demonstration of the critical technology required to service cryogenically cooled satellites in a micro-gravity environment. Superfluid helium has a number of unusual properties including zero viscosity, very high thermal conductivity, and a thermo-mechanical effect in which the helium is "attracted" to heat. The experiment will be controlled continuously during a four to seven day mission by a scientist at a Payload Operations and Control Center (POCC) or by an astronaut on the Aft Flight Deck (AFD) of the shuttle orbiter. Mission objectives for SHOOT include demonstrating the transfer of superfluid helium between two dewars, verifying the design of a superfluid helium transfer line and man rated fluid coupler, demonstrating a technique called heat pulse mass gauging to very accurately determine the amount of helium present and the onboard control of a helium transfer without ground intervention. All of these are necessary for the future development of NASA's Superfluid Helium Tanker (SFHT) which will NASA's next generation of astrophysical observatories in low earth orbit (i.e., SIRTF, AXAF, ASTROMAG).

[§] Sterling Software, Palo Alto, California

Some of these objectives are best accomplished by an expert operator from the POCC, others must be accomplished by a shuttle crewman having real time visibility and control of SHOOT systems. As a result, the operations and control software has been divided into two separate and distinct parts. Each has its own capabilities tailored for the environment, specific operations, and level of expertise of the operator. POCC software is designed to preserve the maximum operational flexibility for an expert operator during all phases of SHOOT hardware use including pre, post and in-flight experiments. AFD software is designed to provide a non-expert shuttle crewmember with the specific information and control functions for a given experiment phase. Diagnostics and error handling are provided for off nominal conditions.

Experiment Overview

The SHOOT experiment consists of a series of transfers between two dewars in the shuttle cargo-bay. Each dewar (Figure 1.) is heavily instrumented with temperature, pressure and fluid level sensing devices. Operators at the POCC can select individual devices to read and display through commands sent to the SHOOT electronics on the shuttle (Figure 2.). Data that is received at the POCC is converted to meaningful units, limit checked and archived in real time. In this way the experiment control computer at the POCC is the experiment PI's interface to the flight hardware. Most operations are performed under POCC control.

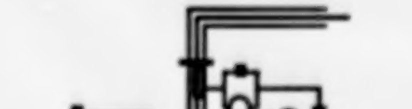


Figure 1. The SHOOT dewar and cryostat

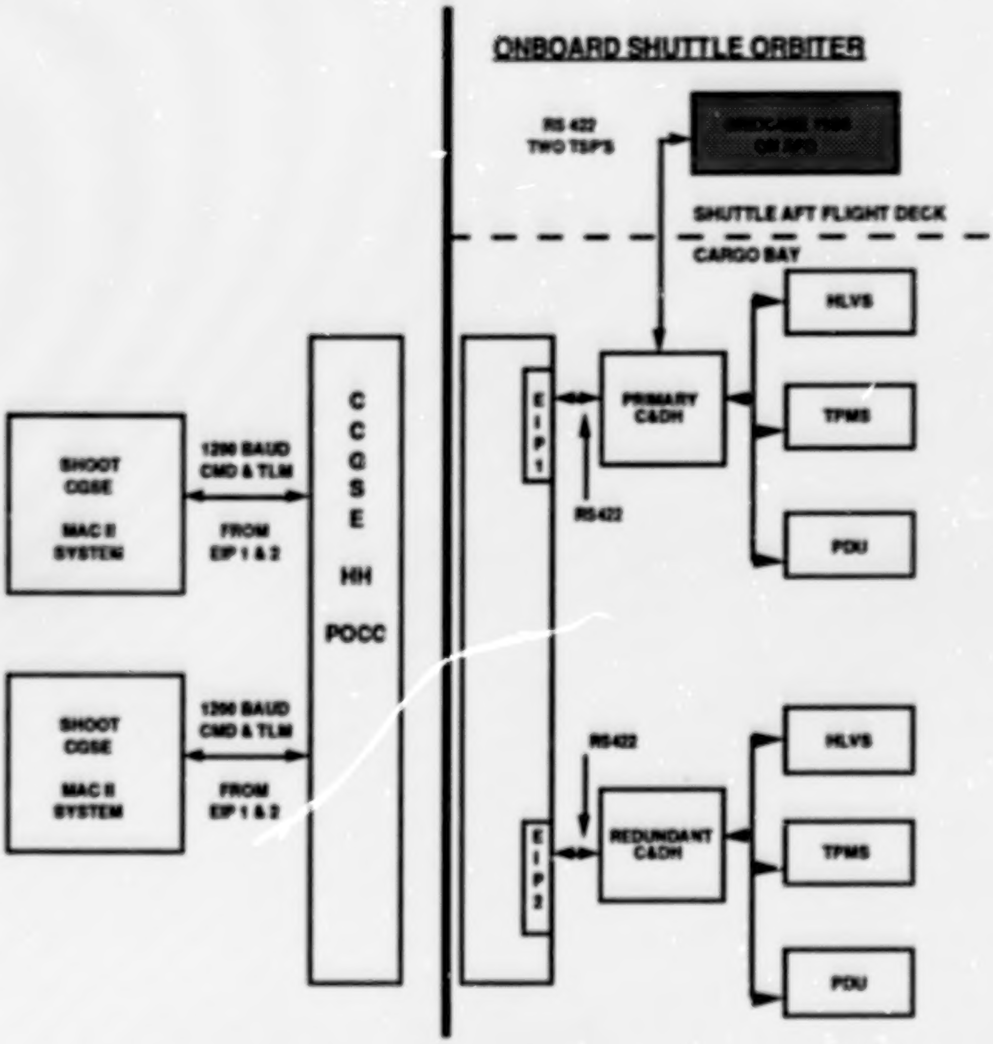
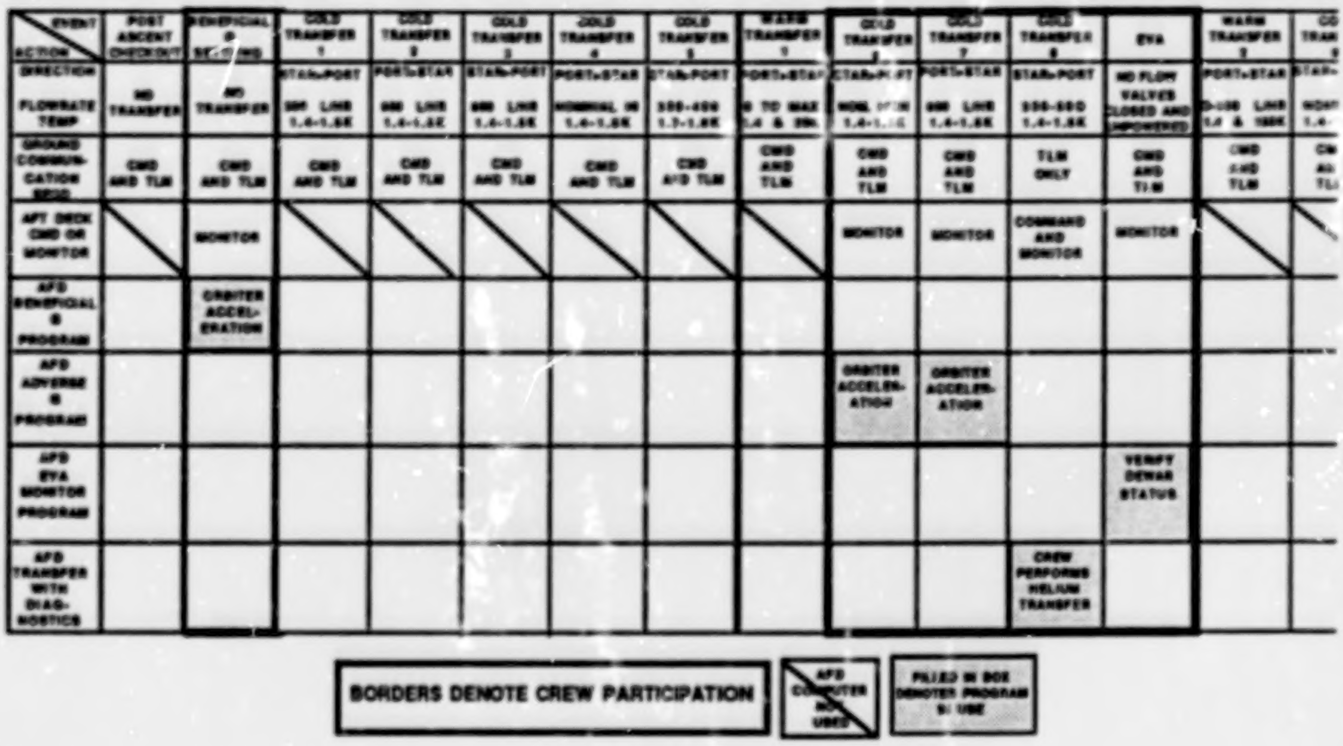

ONBOARD SHUTTLE ORBITER
RS 422
TWO TSP'S
SHUTTLE AFT FLIGHT DECK
CARGO BAY
HLVS
TPMS
PDU
PRIMARY C&DH
EIP 1
RS422
SHOOT CGSE
MAC II SYSTEM
1200 BAUD CMD & TLM
FROM EIP 1 & 2
C C G S E
HH
POCC
SHOOT CGSE
MAC II SYSTEM
1200 BAUD CMD & TLM
FROM EIP 1 & 2
RS422
EIP 2
REDUNDANT C&DH
HLVS
TPMS
PDU

Figure 2. Block Diagram of the SHOOT Electronics

Legend: C&DH-Command and Data Handling Unit, HLVS-Heater, Level Detector Valve Driver System, TPMS-Temperature and Pressure Measurement System, CCGSE-Customer Carrier Ground Support Equipment, EIP-Experiment Interface Panel

Certain operations require testing the ability of the SHOOT Fluid Acquisition apparatus to pump helium during normal orbiter attitude maintenance maneuvers. Since the SFHT will be shuttle based verifying this capability is of prime importance to achieving the SHOOT science objectives. This involves real-time, graphical feedback to the crew during Reaction Control System jet firings. The crew will initiate the firing during a helium transfer and observe the system performance in real time. Upon observation of a loss of flow condition the acceleration is to be terminated and the ability of the flow to restart observed. Other operations such as the Beneficial Acceleration Settling and the Extra-Vehicular Activity (EVA) Monitoring function are also best performed onboard. Monitoring programs specific to these functions will be provided for the crew's use.

BEST COPY AVAILABLE



EVENT / ACTION	POST ASCENT CHECKOUT	BENEFICIAL g SEEKING	COLD TRANSFER 1	COLD TRANSFER 2	COLD TRANSFER 3	COLD TRANSFER 4	COLD TRANSFER 5	WARM TRANSFER 1	COLD TRANSFER 6	COLD TRANSFER 7	COLD TRANSFER 8	EVA	WARM TRANSFER 2	CO TRA
DIRECTION FLOWRATE TEMP	NO TRANSFER	NO TRANSFER	STAR→PORT 500 L/HR 1.4-1.8K	PORT→STAR 800 L/HR 1.4-1.8K	STAR→PORT 800 L/HR 1.4-1.8K	PORT→STAR NOMINAL IN 1.4-1.8K	STAR→PORT 300-400 1.7-1.8K	PORT→STAR 0 TO MAX 1.4 & 2.0K	STAR→PORT NOM. [illegible] 1.4-[illegible]	PORT→STAR 800 L/HR 1.4-1.8K	STAR→PORT 300-500 1.4-1.8K	NO FLOW VALVES CLOSED AND UNPOWERED	PORT→STAR 0-100 L/HR 1.4 & 100K	STAR→ NOM 1.4-
GROUND COMMUNICATION SPSD	CMD AND TLM	CMD AND TLM	CMD AND TLM	CMD AND TLM	CMD AND TLM	CMD AND TLM	CMD AND TLM	CMD AND TLM	CMD AND TLM	CMD AND TLM	TLM ONLY	CMD AND TLM	CMD AND TLM	CM AN TL
AFT DECK CMD OR MONITOR	╲	MONITOR	╲	╲	╲	╲	╲	╲	MONITOR	MONITOR	COMMAND AND MONITOR	MONITOR	╲	╲
AFD BENEFICIAL g PROGRAM		ORBITER ACCELERATION												
AFD ADVERSE g PROGRAM									ORBITER ACCELERATION	ORBITER ACCELERATION				
AFD EVA MONITOR PROGRAM												VERIFY DEWAR STATUS		
AFD TRANSFER WITH DIAGNOSTICS											CREW PERFORMS HELIUM TRANSFER			

BORDERS DENOTE CREW PARTICIPATION | ╲ AFD COMPUTER NOT USED | ▨ FILLED IN BOX DENOTES PROGRAM IN USE

Figure 3. Matrix showing the division of control between the ground and crew

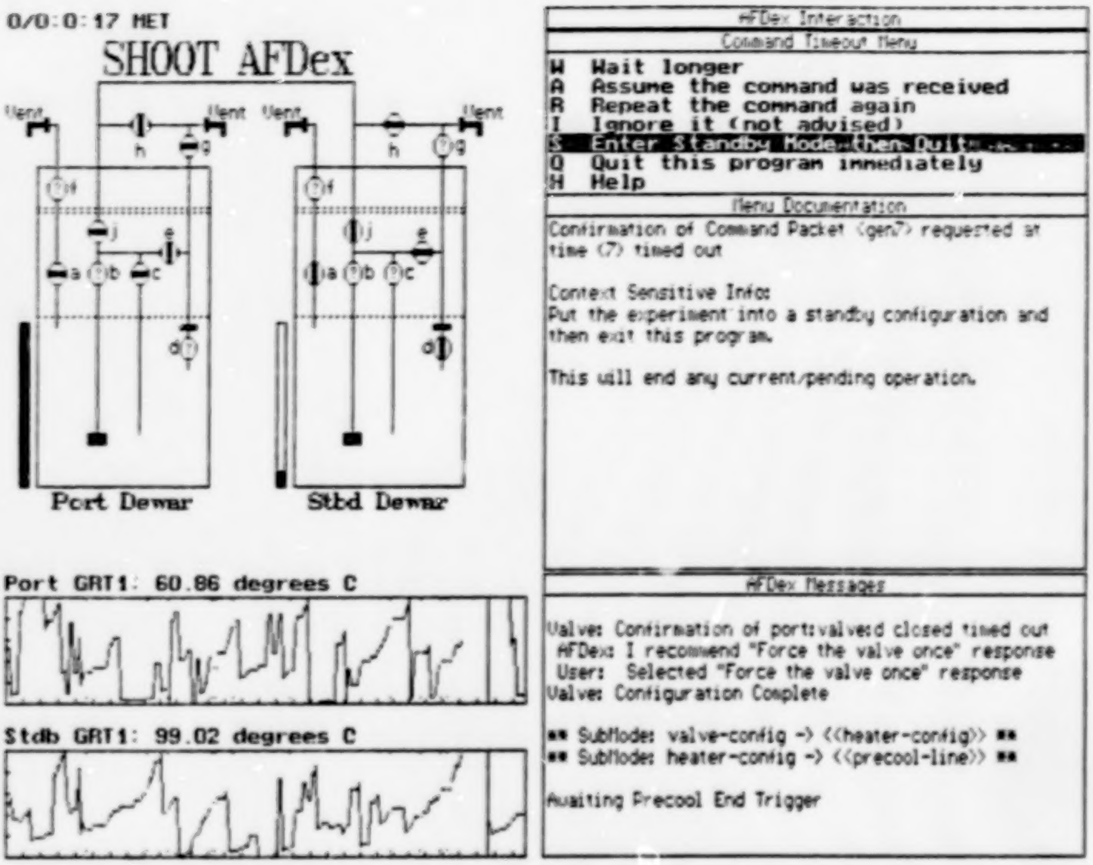
The SFHT will contain many thousands of liters of helium if it is to meet the requirements of large space based observatories. Fluid transfer rates up to 1000 liters per hour are conceivable. Even at this flow rate a satellite resupply is a multi-hour process. The objective of the AFD expert system is to demonstrate the ability to control a helium transfer operation autonomously from orbit without crew or ground intervention.

Aft Flight Deck Software

Four programs are being developed to support SHOOT science objectives from the Aft Flight Deck. Three of these are for monitoring only, while the fourth will demonstrate the autonomous control of cryogen transfer in space without ground intervention.

AFDex (\'af-dek\) is a rule-based system under development at the NASA Ames Research Center for the SHOOT experiment, which will operate on the Space Shuttle's Aft Flight Deck (AFD) computer. The primary goal of AFDex is to provide intelligent process control, diagnosis, and error recovery capabilities for the transfer of superfluid helium between two dewars in the orbiter's cargo bay. During a nominal transfer, AFDex is responsible for sending commands which control the payload, monitoring telemetry from the payload, and providing a graphical display which reflects the current state of the dewars and the transfer. In the event of an abnormal condition, AFDex must diagnose the condition and formulate plans for error recovery. Diagnosis is associative (based upon relationships between symptoms and causes) as opposed to a model-based approach which would be too inefficient for this application.

BEST COPY AVAILABLE


0/0:0:17 MET
SHOOT AFDex
Vent
Vent
Vent
Vent
h
g
h
g
f
f
j
e
j
e
a
b
c
a
b
c
d
d
Port Dewar
Stbd Dewar
Port GRT1: 60.86 degrees C
Stdb GRT1: 99.02 degrees C
AFDex Interaction
Command Timeout Menu
W Wait longer
A Assume the command was received
R Repeat the command again
I Ignore it (not advised)
S Enter Standby Mode then Quit
Q Quit this program immediately
H Help
Menu Documentation
AFDex Messages

```
Confirmation of Command Packet <gen7> requested at
time (7) timed out

Context Sensitive Info:
Put the experiment into a standby configuration and
then exit this program.

This will end any current/pending operation.
```

```
Valve: Confirmation of port:valve:d closed timed out
 AFDex: I recommend "Force the valve once" response
 User:  Selected "Force the valve once" response
Valve: Configuration Complete

** SubMode: valve-config -> <<heater-config>> **
** SubMode: heater-config -> <<precool-line>> **

Awaiting Precool End Trigger
```

Figure 4. Display of AFDex during Precool Phase

Although the system is designed to automate the transfer operation, AFDex has facilities to interact with the astronauts. Data which is relevant to a current operation is displayed graphically. An astronaut may guide the system's behavior (i.e. select an error recovery strategy) via menus and input fields. Input is asynchronous and preemptable. Asynchrony allows the system to continue reasoning about events while interacting with the user. Preemptability allows the system to interrupt a current interaction in response to some event.

AFDex receives data via a serial connection through the Hitchhiker carrier's avionics. Since the system must respond in real-time to to events represented in this data stream, a number of mechanisms have been developed to reduce the effort of processing payload data. The system may be viewed as consisting of three components: the payload avionics, a telemetry preprocessor, and the rule-based system. At the lowest level, AFDex may command the avionics hardware to control what data it receives from the payload and at what rate. This provides a coarse-grained, low overhead filter suitable for specific operations although it has limited flexibility and nontrivial operational complexity.

Once telemetry is received by the Aft Flight Deck computer from the payload avionics, a preprocessor is used to focus attention on certain classes of data (to the exclusion of others) from which it will create facts to assert into the expert system knowledge base. The behavior of this preprocessor is dynamically controlled by the expert system. In an effort to reduce the overhead of processing large amounts of invariant data, the preprocessor may be commanded to report only relevant changes in data values. Additional preprocessing

BEST COPY AVAILABLE

may include qualitative analysis (i.e. from 23 degrees to warm), constraint demons (alert if GRT-12 and GRT-35 are the same temperature), and deviation from anticipated (simulated) values. Data from the preprocessor is asserted into a Rete network [Forgy] which efficiently handles pattern matching and rule invocation. Within the expert system a number of strategies (modalities, prioritized agenda, rule partitioning, etc.) are used to focus attention and respond efficiently to events.

AFDex is being developed in CLIPS (a derivative of OPS-5 and ART developed by NASA) and C for delivery on a 386-based GRiD 1530 laptop computer. The current system has knowledge concerning command generation and receipt, valve devices, germanium resistance thermometers, all operations required for nominal transfer, and simulation of certain error modes for each device. This is captured in a knowledge-base currently consisting of 84 rules and 114 initial facts. The capabilities and size of the system will increase significantly as development continues.

SHOOT utilizes three additional programs on the Aft Flight Deck computer. These programs provide real-time monitoring capabilities during specific SHOOT operations. Currently these programs have no commanding capability and thus are dependent upon ground control. We are currently evaluating the option of embedding these programs within rule-based systems similar to AFDex. This would allow certain mission objectives to be achieved in the event of an extended loss of ground communication.

Knowledge-based process control and diagnosis has proved effective in the development of AFDex. A rule-based approach provides a natural framework to develop a system in which multiple threads of execution are inherent. There are potentially many actions possible at any time, and each action may be only slightly related to any other action. Diagnosis is unobtrusively interleaved with control. Rules provide an excellent mechanism for applying large amounts of very specific knowledge in response to changes in the environment. CLIPS has proved to be a fast, compact, and portable system which is able to represent this knowledge and apply it in real-time. This is complemented by the availability of all source code which allows CLIPS to be customized and embedded in an arbitrary application. More significantly, AFDex demonstrates that expert systems have become a mature technology which is being integrated into aerospace operations.

Shoot Ground Based Command and Monitoring System

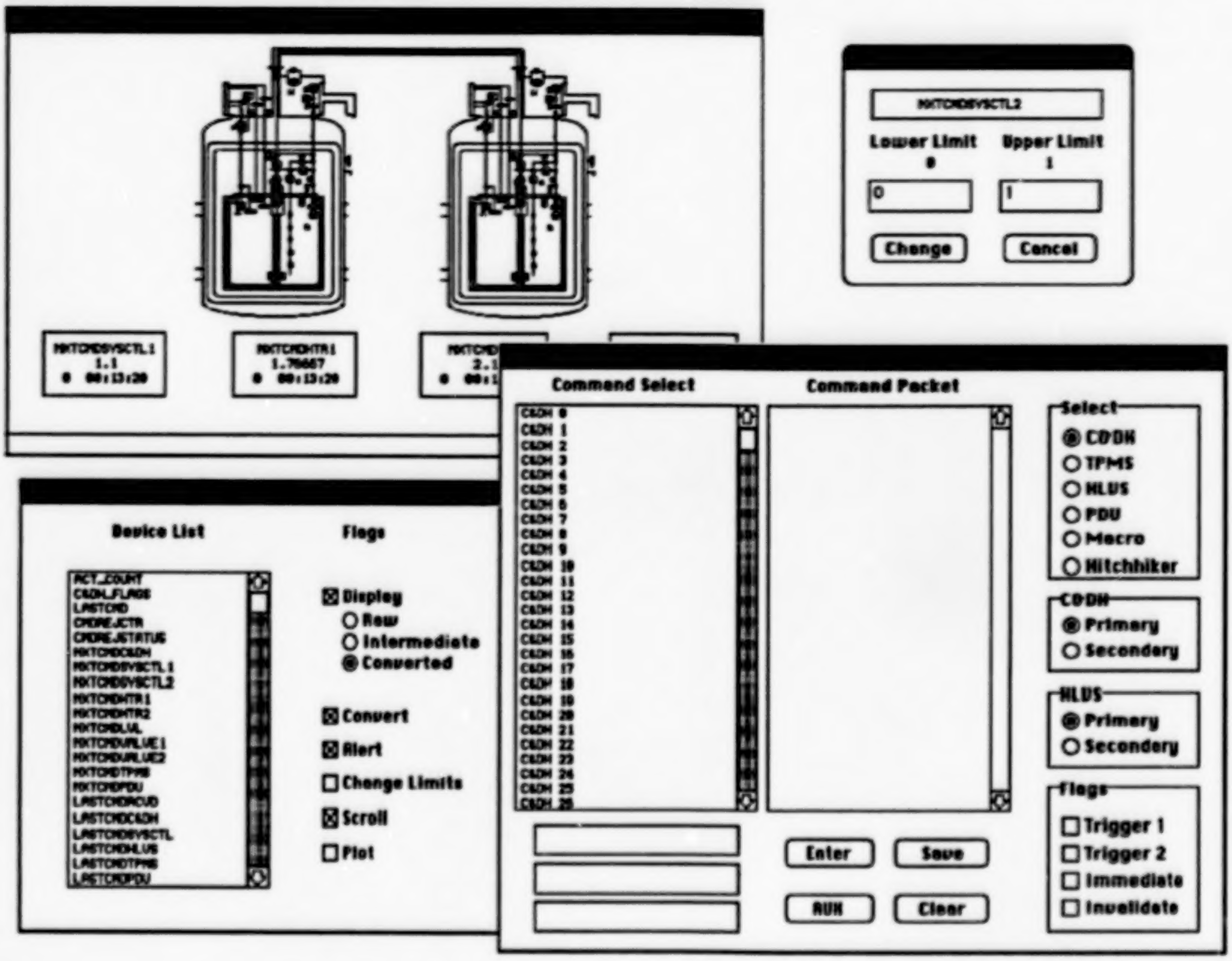
Design

The goal in designing the SHOOT Command and Monitoring System (CMS) software was to create a system that is easy to use as possible while retaining full control of the experiment at a low level for maximum flexibility. Because the SHOOT experiment is highly interactive and will be controlled for extended periods of time, a good user interface is very important. The Macintosh's icon based point-and-click environment provides quick and easy methods for performing complicated tasks.

The user interface of the CMS was fully prototyped before any coding was done and drove the design of the software using a top down approach. Windows were designed from the user's perspective (a specialist in cryogenics) to perform each function required.

Flexibility will allow the software to be reusable for all aspects of the experiment including early tests with the hardware, system level integration, servicing the experiment while in the payload bay of the shuttle on the launch pad, and operations during and after flight.

Should the CMS become overloaded it is designed to degrade gracefully. Every function in the system has a certain priority; lower priority operations will be gradually phased out until higher priority operations have time to catch up. (Figure 5.)


MXTCMDSVSCTL1
1.1
0 00:13:20
MXTCMDHTR1
1.70667
0 00:13:20
MXTCMD
2.1
0 00:1
MXTCMDSVSCTL2
Lower Limit
0
Upper Limit
1
Change
Cancel
Device List
Flags
ACT_COUNT
C&DH_FLAGS
LASTCMD
CMDREJCTR
CMDREJSTATUS
MXTCMDC&DH
MXTCMDSVSCTL1
MXTCMDSVSCTL2
MXTCMDHTR1
MXTCMDHTR2
MXTCMDLUL
MXTCMDVALVE1
MXTCMDVALVE2
MXTCMDTPMS
MXTCMDPDU
LASTCMDACUD
LASTCMDC&DH
LASTCMDSVSCTL
LASTCMDHLUS
LASTCMDTPMS
LASTCMDPDU
Command Select
Command Packet
C&DH 0
C&DH 1
C&DH 2
C&DH 3
C&DH 4
C&DH 5
C&DH 6
C&DH 7
C&DH 8
C&DH 9
C&DH 10
C&DH 11
C&DH 12
C&DH 13
C&DH 14
C&DH 15
C&DH 16
C&DH 17
C&DH 18
C&DH 19
C&DH 20
C&DH 21
C&DH 22
C&DH 23
C&DH 24
C&DH 25
C&DH 26
Enter
Save
RUN
Clear
Select
CDDH
TPMS
HLUS
PDU
Macro
Hitchhiker
CDDH
Primary
Secondary
HLUS
Primary
Secondary
Flags
Trigger 1
Trigger 2
Immediate
Invalidate

☒ Display
○ Raw
○ Intermediate
● Converted

☒ Convert
☒ Alert
☐ Change Limits
☒ Scroll
☐ Plot

Select: ● CDDH ○ TPMS ○ HLUS ○ PDU ○ Macro ○ Hitchhiker

CDDH: ● Primary ○ Secondary

HLUS: ● Primary ○ Secondary

Flags: ☐ Trigger 1 ☐ Trigger 2 ☐ Immediate ☐ Invalidate

Figure 5. Prototype CMS display showing telemetry, command and status windows

Implementation

The SHOOT Command and Monitoring System is being developed on a Macintosh II using the Macintosh Programmers Workshop (MPW) Toolset and is being written in C. The event oriented operating system on the Macintosh enables simultaneous user interaction and internal processing to occur without having multiple processes.

Priority is highest for archiving and calibrating telemetry, followed by commanding operations and updating the display. Two identical systems will be used during flight at the POCC for redundancy, each capable of performing everything required to conduct the experiment. Only one system will be allowed to up-link commands at a time. The other

BEST COPY AVAILABLE

system may be used for more time consuming tasks such as trend analysis of old data or plotting new data as it arrives in real time.

The software is logically broken down into modules corresponding to functionality. Telemetry processing consists of buffering the data as it comes in through the serial port, decommutating a data packet, performing conversions of raw data into engineering units, and archiving both raw and converted data to disk (Figure 6.). The user is notified if any telemetry indicates an error condition.

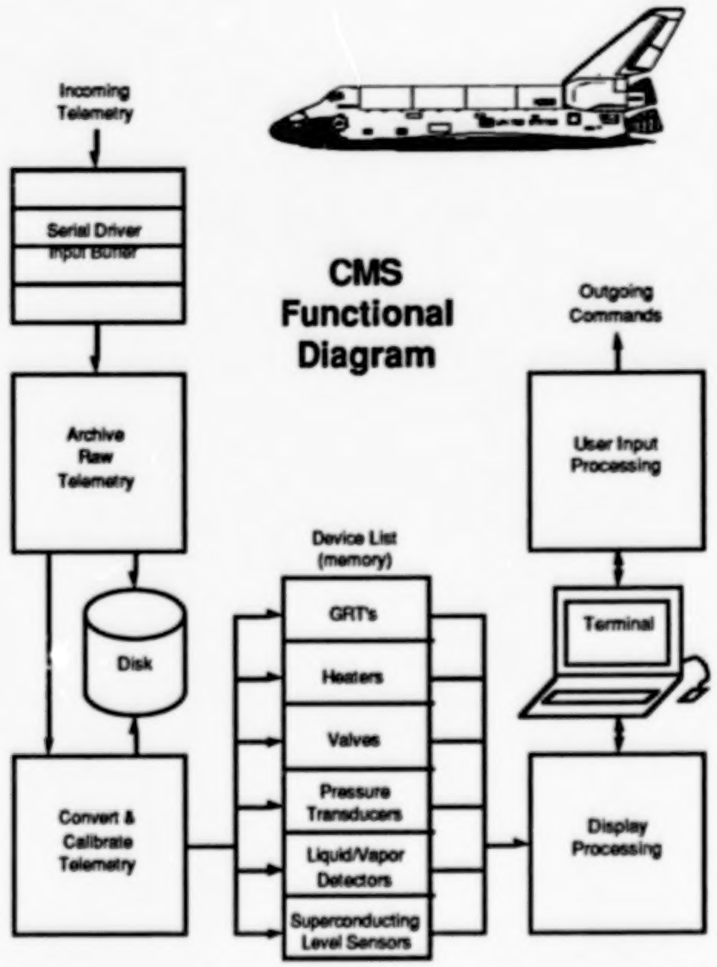

Incoming Telemetry
Serial Driver Input Buffer
CMS Functional Diagram
Outgoing Commands
Archive Raw Telemetry
User Input Processing
Device List (memory)
GRT's
Heaters
Valves
Pressure Transducers
Liquid/Vapor Detectors
Superconducting Level Sensors
Disk
Terminal
Convert & Calibrate Telemetry
Display Processing

Figure 6. Functional diagram of the ground based CMS software

Command processing consists of a portion of the user interface to construct command packets and a command assembler which formats the packet for up-link and sends it out the serial port. Commanding can be automated for repetitive operations and a macro facility exists to simplify the creation of a packet.

Display processing is responsible for updating the screen and for changing display attributes. The telemetry being displayed is disjoint from telemetry processing so that old data may be viewed even if it is not present in the current down-link packet, new data is not necessarily being displayed, although it is still being checked for error conditions and archived. The display for a specific device is updated as new corresponding data comes in.

BEST COPY AVAILABLE

Each device may be individually selected for numerical display and a subset of all devices is always displayed graphically. Alerts and other error conditions are made known to the user in both the telemetry display window and a separate alert window. In addition an audible bell may sound if desired.

The user interface also allows general configuration of the system as well as facilities for performing utility functions such as heat pulse mass gauging or integrating the flow rate.

The development of the SHOOT Command and Monitoring System on a widely available and easy to use system, the Macintosh II, is proving to be both cost effective and efficient. The most important features are the point-and-click interface for interactive control and the reusable nature of the software.

Conclusions

A knowledge-based approach facilitates the use of complex hardware by novice user. Point-and-click interfaces provide an expert user with powerful, intuitive, and flexible control over the same hardware. SHOOT represents the incorporation of these mature technologies into aerospace operations.

Superfluid Helium On-Orbit Transfer is a NASA Office of Space Flight Advanced Program Development Branch sponsored flight experiment managed by the Goddard Space Flight Center Cryogenics Technology Branch of the Engineering Directorate. The Ames Research Center Artificial Intelligence Research Branch of the Information Sciences Directorate is supporting GSFC by developing the software.

References

Forgy, C.L. "On the Efficient Implementation of Production Systems", Ph.D. dissertation, Carnegie-Mellon University, 1974

BLANK PAGE

Shared Resource Control between Human and Computer

James Hendler*
Reid Wilson
Computer Science Dept.
University of Maryland, College Park

February 6, 1989

1 Introduction

One area of Artificial Intelligence systems for space applications is to serve as a mechanism for telerobotic control. When interaction is called for, it is usually viewed as a passing of control with *either* an AI system *or* a human in complete command of the teleoperated system. Unfortunately, this can cause serious inefficiencies in the planning process, both for the AI system and the human, when the plans and actions of each may cause potential conflicts. If the human were to, for example, remove a bolt which the AI system had previously placed, then it is important for the robot to take this information into account in planning a later action concerning that bolt or its assembly. If this information must be rediscovered by the AI system upon regaining control, the system must spend much time both checking for states already achieved and for replanning the accomplishment of previously achieved actions.

This paper describes the advantages to an AI system of actively monitoring a human's control of a shared resource (such as a telerobotic manipulator). We describe a system in which a simple AI planning program gains efficiency by monitoring a human's actions and recognizing when they cause a change in the system's assumed state of the world. This enables the planner to recognize when an interaction occurs between the human's actions and the system's goals, and allows it to maintain an up-to-date knowledge of the state of the world and thus to (i) inform the human when an action would undo a goal achieved by the system, (ii) inform the human when an action would render a system goal unachievable, and (iii) efficiently replan the establishment of goals after human intervention.

---

*Funding for this work was provided in part by the Office of Naval Research, the UM Institute for Advanced Computer Studies, and the Systems Research Center

2 Shared Resource Control

The current view of shared resource control between a human and an AI system generally centers on one or the other partner being in full control of the resource at any given time. Thus, in a telerobotic system, either a human operator or an AI system may be controlling an arm. When the human is in control the AI system is quiescent. However, when the AI system is in control, the human is viewed as being in a supervisory role. This allows a human to monitor the system and make sure that no unanticipated problems are caused by the system. (This is frequently the case in telerobotic systems because of the need for a human operator to deal with unanticipated or emergency situations outside of the AI system's problem solving capabilities. This is particularly important when the situation is somewhat fluid, that is, when an earlier unexpected event may cause repurcussions not anticipated at system design time. As an example of such an event, consider an undeployed solar collector which might cause power consumption considerations not anticipated at system design time).

It is our contention that in a similar manner to a human's supervision of the AI system, the AI system must monitor the human's use of the shared resource. While we believe that is is important that the human maintains control (i.e. has the final word), the AI component can both gain efficiency and provide some measure of error checking by performing this monitoring. Allowing the AI system to monitor human performance avoids serious inefficiencies in the planning process and provides benefits not possible where the human is not monitored.

Probably the most important benefit to the AI system is its ability to update its internal view of the world to reflect any changes observed during the times of human control. This is very important, because by monitoring, the system will always have an accurate view of the world. Although there is some overhead in such a process, the planner will have to update its state of the world in response to each human action, the benefit provided is that the AI system never needs to reestablish the current state of the world.

The ability of the system to maintain a complete and accurate world model is quite important. In current systems, upon completion of the human's control of the resource, the AI system is left with an incomplete, and possibly incorrect world model. Thus, for example, if a human has moved some object (say a bolt), the robot must either explicitely check that the object is still where it was previous to human intervention (requiring a move to the location of the bolt), or it must assume no harmful interactions will have been caused by the human and try to work with an incomplete state model. Unfortunately, working with an incomplete state model is a major problem for current AI system (a long discussion and literature review of this issue is presented in Sanborn and Hendler, 1988). Thus, the planner is faced by a choice between reestablishing all conditions, which is a highly resource intensive and inefficient process, or it must work from incomplete knowledge with all the attendant dangers. By monitoring the human's actions, to maintain an awareness of the actual state of the world, the planner need not have to rediscover information each time control is returned to it without having to sacrifice information certainty.

Another reason for the need for the AI system monitoring the human is that when partners share control of a resource they may have conflicting, complementary, and/or unrelated goals. In each case, having the passive agent monitor the actions of the controlling partner allows significant benefits:

- Where a conflict arises, the partners may resolve it early.
- Where cooperation may occur, similar preconditions are not reestablished needlessly.
- Where goals are unrelated, accidental interactions (such as moving a tool required later) are largely alleviated.

In the remainder of this paper we describe a small system which was designed solely to show the efficacy of such monitoring. In the next section we describe the system and show, by way of an example, how the monitoring provides for the sorts of interactions described above. This system is quite limited, and is in no way proposed as an actual telerobotic planning system. Some steps towards that direction are discussed in section 4.

3 An Example System

To demonstrate how shared control better enables the partners to handle interacting goals, we have developed a very simple blocks world system in which control of a "robot" is shared between a human operator and an Artificial Intelligence Planning system. (In section 4 we will discuss limitations of this sort of system viz. realistic telerobotic control.) Instead of having one or the other partner control the robot independently (i.e., the other not receiving simultaneous feedback), both parties are active in monitoring the actions of the other and the effects on their goals. Thus this shared control allows the passive agent will determine how the actions of the controlling agent affect its goals, and react appropriately (interacting as necessary).

In our system the "robot" can be used for the manipulation of items in a simple domain: a set of various blocks of different size and orientations. Neither the blocks nor the robot may be moved off the edge a simple "world" consisting of a simple grid. Only one agent (human or computer) may actively control the robot at one time, although the other receives the commands given to the robot before they are executed. While the AI system is in control, the human is allowed the human to assume control of the robot at any time (essentially, after any discrete move or step).

A set of commands is given to an AI planning system, and it attempts to control the "robot" to acheive these goals. To do this, the planner maintains three (3) goal stacks: goals to do (TO-DO), goals done (DONE), and goals the planner needs to re-establish (RE-DO) because the human broke them after they were achieved. The planner does not sort the TO-DO or RE-DO goal stacks, but attempts to complete them in order from the top goal in the stack to the bottom. Thus, we assume that a goal ordering has already been achieved earlier in the planning cycle, and that we are now at execution time.

The AI system is designed to assume that a human is better able to determine how to deal with difficulties in the world. Thus, If the current goal cannot be achieved, the planner prompts

```
  0123456789             Goals:
0 ----------               TO-DO: ((BLOCK1 ((0 2)) AT))
1 -23-------               RE-DO: NIL
2 -13R------               DONE:  NIL
3 -43-------
4 ----------
5 ----------
6 ----------
7 ----------
8 ----------
9 ----------

: AI system cannot achieve move of BLOCK1 to (0 2). Please assume
: control.
```

Figure 1: AI system requests human intervention.

the human for help to achieve it. Figure 1 demonstrates a case in which the AI system can accomplish a goal with the human's assistance that it could not achieve on its own. A deficiency of the planner is that it does not know it can push a block into its goal position by pushing another block into it. When it recognizes its goal is unachievable, it prompts the human for intervention. The human may then evaluate whether the goal should be modified or assume control of the system to help the planner achieve it. In this example, the human can just push BLOCK3 left or move it out of the way. With the human's assistance, complicated goals may be reduced to simpler goals achievable by the planner.

A more interesting behavior can be observed when an interaction occurs between the goals achieved by the planning system and the actions required by the human. In this case, the robot warns the human about the potential interaction, and asks for confirmation. Thus, the human is able to avoid accidently interfering with the planning system. If another move would solve the same problem, the human might choose to avoid the interaction.

In Figure 2, we see a situation where trying to move the robot to the right will result in the human breaking a achieved goal namely the origin of BLOCK1 at position (7 7). The AI planner, monitoring the human's actions, will require confirmation that this is an intentional consequence. In this case, the human will avoid breaking the goal. Checking the goal stacks, one sees that no change has occurred.

If the human does decide to proceed with the move, the planner is able to recognize that a goal has been made invalid (i.e. "broken"). This goal is moved to a RE-DO stack, and will be re-established when the planner regains control. In this manner, the planning system is able to avoid taking later steps which rely on previously achieved goals which are no longer valid.

```
  0123456789
0 ----------
1 ----------
2 ----------
3 ----------
4 ----------
5 ----------
6 ----------
7 ------R11-
8 --2----333
9 ----------

Current Goals:      TO-DO: ((BLOCK2 ((7 6)) AT) (BLOCK3 ((7 8)) AT))
                    RE-DO: NIL
                    DONE:  ((BLOCK1 ((7 7)) AT))
Human Action:       move right
System Response:    Moving to square (5 4) will break an achieved goal.
                    Are you sure? (Y/[N])
Human Action:       N
Current Goals:      TO-DO: ((BLOCK2 ((7 6)) AT) (BLOCK3 ((7 8)) AT))
                    RE-DO: NIL
                    DONE:  ((BLOCK1 ((7 7)) AT))
```

Figure 2: AI makes human aware of interaction

In Figure 3, we see that the human is not directly pushing a block from a required location, but as a consequence of the command to push BLOCK3 the already completed goal involving BLOCK1 will be broken. Prompted again for confirmation, the operator this time confirms the action. At this point we see that in the planners goal tables, the BLOCK1 goal has been moved from the DONE stack to the RE-DO stack. Because the AI system monitored the human's actions, it realizes that goal needs to be re-achieved, and plans accordingly. (Note that the goal involving BLOCK3, which was at its target destination, was not moved because the planner only sees the top goal in the stack, and had not therefore achieved that goal.)

In addition, beneficial actions may also be observed. This, if the AI system recognizes that the human has moved the robot such that the planner's current goal is achieved, the system is aware of this and moves the goal to the DONE stack. Had the human not re-achieved the goal, that would have become the planner's current goal.

Finally, to remphasize, at each stage in the interaction the AI planning system is maintaining an accurate world model. goal. Because it has been monitoring the human's actions, it knows where the robot is, where each block is, and the correct contents of the goal stacks. Thus, upon

```
  0123456789
0 ----------
1 ----------
2 ----------
3 ----------
4 ----------
5 ----------
6 ----------
7 -------11-
8 --2----333
9 -------R--

Current Goals:     TO-DO: ((BLOCK2 ((7 6)) AT) (BLOCK3 ((7 8)) AT))
                   RE-DO: NIL
                   DONE:  ((BLOCK1 ((7 7)) AT))
Human Action:      move up
System Response:   Moving to square (5 4) will break an achieved goal.
                   Are you sure? (Y/[N])
Human Action:      Y
Current Goals:     TO-DO: ((BLOCK2 ((7 6)) AT) (BLOCK3 ((7 8)) AT))
                   RE-DO: ((BLOCK1 ((7 7)) AT))
                   DONE:  NIL
```

Figure 3: Goal moved to RE-DO stack

regaining control it plans its current goal without having to resetablish the positions of the blocks.

4 Real-time Considerations

While the simple system we have implemented shows the need for the AI system to monitor human operation, in the real world the problem is obviously significantly more complex. The most significant difference is that the human control is *not* performed by giving easily recognizable commands to an AI system, but rather by the continuous operation of the telerobotic device. Thus, even the simple recognition of the human's invalidating of a previously achieved goal becomes a major problem.

For monitoring to be applicable in the real world, as opposed to a simulation environment, a large amount of run-time support for real-time computing is necessary. Our current research (Hendler, 1989) focuses on examining shared resource control in a real-time environement via the use of MARUTI, a hard real-time operating system developed by the Systems Research Group at the university of Maryland (Levi et. al, 1988). The MARUTI OS is designed to support real-time applications on a variety of hardware platforms. MARUTI supports guaranteed-service scheduling, in which jobs that are accepted by the system are verified to satisy general time constraints. In MARUTI, the system automatically verifies the schedulability of each component of a job with respects to its constraints and those of other jobs in the system.

The use of real-time in a telerobotic environment also poses a problem. The tolerance for response times must be large enough that the human may effectively interact, and without obsolete information. However, when there is a time lag, as in space applications, having the AI system monitor the human actions may be especially useful as the human might not otherwise get feedback until damage is already done. One does not, for example, wish an operator to get a warning about a potential interaction after it has occurred.

5 Conclusions

In this paper we have discussed the need for having an AI system monitor human usage of a shared resource. As discussed, we have demonstrated that this allows greater efficiency for the AI system, as an accurate world model can be maintained, and also allows for early detection of interactions enabling the discovery of cooperation or the avoidance of conflicts.

6 References

Hendler, J.A. "Real Time Planning" *Proceedings AAAI Spring Symposium on Planning and Search*, March, 1989 (In Press).

Levi, S., Tripathi, S., Carson, S. and Agrawala, A *The MARUTI Hard Real-Time Operating System* Technical Report, Computer Science Dept., University of Maryland, 1988.

Sanborn, J. and Hendler, J. "Monitoring and Reacting: Planning in dynamic domains" *International Journal of AI and Engineering,* Volume 3(2), April, 1988. *p*.95.

BLANK PAGE

An English Language Interface For Constrained Domains

Brenda J. Page
Unisys Corporation
4325 Forbes Boulevard
Lanham, Maryland 20706

The Multi-Satellite Operations Control Center (MSOCC) Jargon Interpreter (MJI) demonstrates an English language interface for a constrained domain. A constrained domain is defined as one with a small and well delineated set of actions and objects. The set of actions chosen for the MJI is from the domain of MSOCC Applications Executive (MAE) Systems Test and Operations Language (STOL) directives and contains directives for signing a crt on or off, calling up or clearing a display page, starting or stopping a procedure, and controlling history recording. The set of objects chosen consists of crts, display pages, STOL procedures, and history files. Translation from English sentences to STOL directives is done in two phases. In the first phase, an augmented transition net (ATN) parser and dictionary are used for determining grammatically correct parsings of input sentences. In the second phase, grammatically typed sentences are submitted to a forward-chaining rule-based system for interpretation and translation into equivalent MAE STOL directives. Tests of the MJI show that it is able to translate individual clearly stated sentences into the subset of directives selected for the prototype. This approach to an English language interface may be used for similarly constrained situations by modifying the MJI's dictionary and rules to reflect the change of domain.

The work described in this paper was done under contract to the National Aeronautics and Space Administration Goddard Control Center Systems Branch.

Introduction

One of the issues in the Multi-Satellite Operations Control Center (MSOCC) environment is the existence of multiple System Test and Operations Language (STOL) dialects. The implementations of STOL in the various MSOCC control centers, such as the past developments of the Earth Radiation Budget Satellite (ERBS) and the Solar Maximum Mission (SMM) projects and the recent developments of the MSOCC Applications Executive (MAE) for the Cosmic Background Explorer (COBE) and Gamma Ray Observatory (GRO) projects and the Data Operations Control System (DOCS) project, have been done independently giving rise to different STOL dialects. The existence of multiple STOL dialects prevents MSOCC users from easily manipulating multiple systems. Artificial intelligence (AI) techniques, such as augmented transition nets (ATNs) and rule-based systems, have been used to build a prototype for a man-machine interface (MMI) common to all MSOCC systems. This prototype, called the MSOCC Jargon Interpreter (MJI), is an English language interpreter. The objective of the MJI is to demonstrate a solution which can alleviate the user's need to know the syntax of any STOL dialect and allow him to concentrate on directing the system to perform the functions for which it was designed.

Since the same activities are preformed over and over in MSOCC, a subset of English language, which we have called MSOCC jargon, is used over and over again to describe those actions. Some of the common actions include starting or stopping objects such as crts, display pages, STOL procedures, and history files. For the prototype, the domain is constrained to the STOL directives (STOL, PAGE, START, KILLPROC, and HISTORY) to perform these actions. The STOL dialect produced by the MJI is the one common to COBE, GRO, and other projects based on MAE.

The MJI contains three major components (figure 1). The first part of the MJI is a context free grammar (cfg) ATN parser which accepts and parses imperative sentences. Sentences are checked by the parser to make sure they are grammatically correct. The parser also identifies sentences' verbs, direct objects, and prepositional phrases and types each word in the sentences as a verb, determiner, adjective, noun, preposition, or conjunction. A dictionary lists the words accepted by the MJI and the parts of speech they can fill.

The second component of the MJI is a rule-based system. Facts from typed sentences are placed into a working knowledge base. An inference engine uses the working knowledge base and sets of rules to determine the meanings of sentences and produce equivalent STOL directives.

The third component of the MJI is an explanation system which allows the user to see the way sentences are grammatically typed by the parser and which rules are used by the inference engine to create STOL directives.

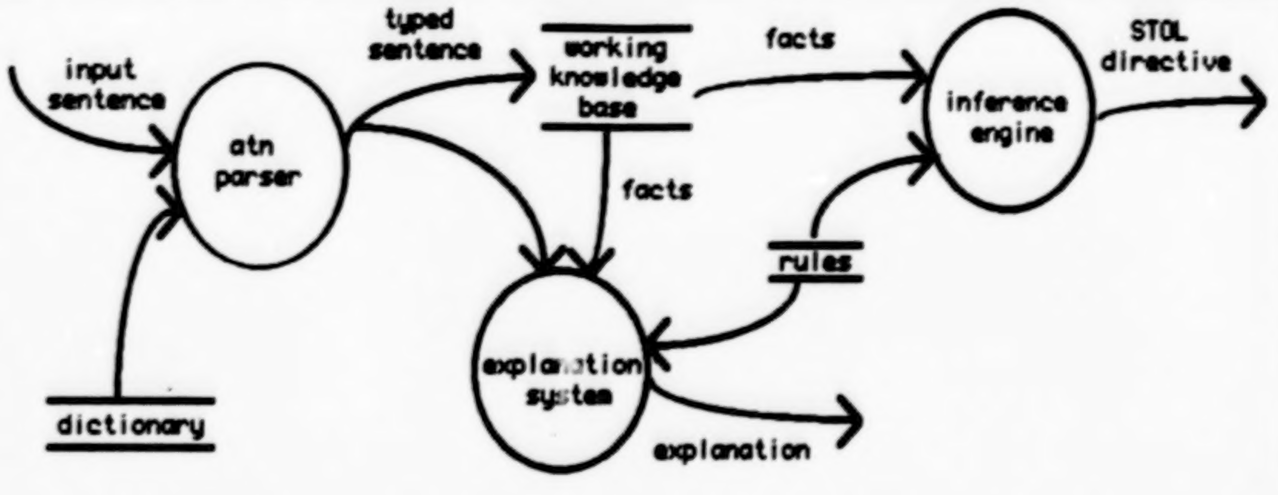


Figure 1. MJI components

The MJI is divided into three parts. The first is an ATN parser which accepts input sentences and grammatically types each word according to its usage (verb, noun, etc.) in the sentence. Typed sentences are then sent to an inference engine which uses rules and a working knowledge base to produce STOL directives. The explanation system allows the user to see the way sentences are grammatically typed and which rules are used to create STOL directives.

The parser

Because of the semantics of the MSOCC environment, only present tense imperative sentences containing a verb, a direct object target of the verb, and prepositional phrases modifying the action taken on the direct object are accepted by the parser. This restriction supports the type of work done in MSOCC. Commands to a system are made with the expectation that they will be carried out immediately. There is no stated subject in the imperative sentences accepted by the prototype - the subject is implied to be the target system.

The knowledge representation scheme chosen to represent context free imperative sentences is an ATN in conjunction with a dictionary. For grammatically typed sentences produced by the ATN, the knowledge representation scheme chosen is frames. Frames contain slots for values and are placed into a hierarchy.

An ATN is a set of states and transitions between the states used to parse an input string to determine whether or not the string is legal under the grammar defined by the ATN. The knowledge of the construct of a legal sentence is kept in the form of the ATN. Figure 2 shows part of the ATN used in the MJI. ATNs differ from traditional nets, which

BLANK PAGE

also consist of stated and transitions, in that they have the capability to take notes as they are traversed and refer to the notes as the traversal continues or when it is done. The note taking feature is used to fill frames containing a sentence's structure of verb, direct object and prepositional phrases.

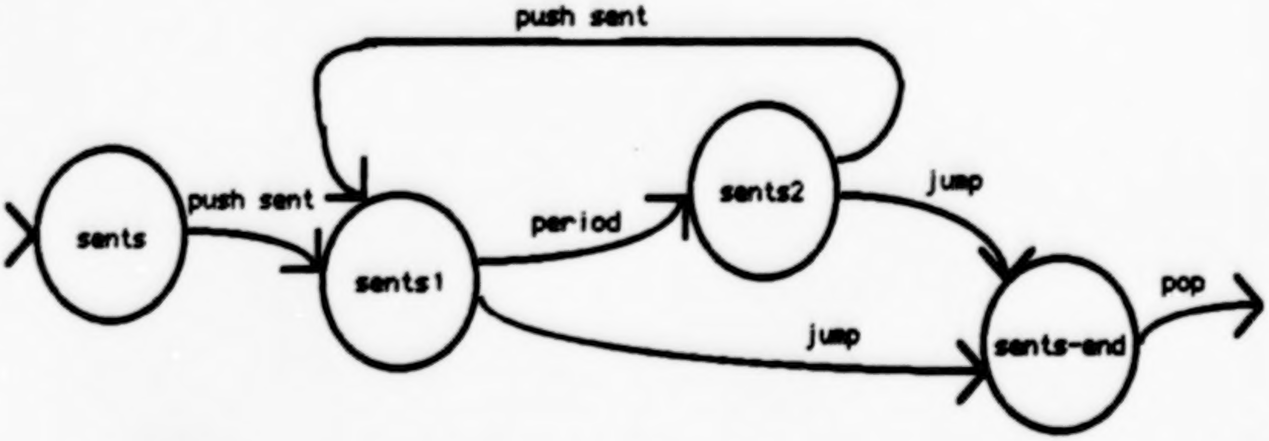


Figure 2. ATN nodes

The sentences ATN allows multiple sentences to be entered separated by periods. Each individual sentence is checked by the sentence ATN via the "push sent" transition. When all sentences are processed, a jump is made to sents-end which returns via a pop. Table 1 shows the definitions for these nodes and figure 3 shows part of the sentence ATN.

An ATN compiler is used to compile ATN node definitions (table 1) into code, each node becoming a procedure. The ATNs are compiled such that traversal proceeds in a depth first search. When a node is executed, it places all next nodes which can legally be reached on the front of a queue. A control mechanism pops the next node off the queue so it can be executed.

The ordering of links leaving a state plays an important role in determining the efficiency of the parser. The links are ordered so the one most likely to lead to a solution is checked first. For example, in the sentence ATN (figure 3) the first part of a sentence can be either a verb or a prepositional phrase. Since the occurrence of imperative sentences beginning with a prepositional phrase is predicted to be less than imperatives beginning with a verb, the verb path is always checked first.

BEST COPY AVAILABLE

Table 1. ATN nodes

The ATN compiler processes each ATN node definition creating procedures. Figure 2 shows the ATN corresponding to these definitions.

```
(atn-node sents
      (push sent sents1))
            ; Push sent ATN so it may be traversed.
(atn-node sents1
      (cat period t sents2 nil)
            ; If the next word is a period, consume the word and move to sents2.
      (jump sents-end))
            ; Move to sents-end.

(atn-node sents2
      (push sent sents1)
            ; Push sent ATN so it may be traversed.
      (jump sents-end))
            ; Move to sents-end.

(atn-node sents-end
      (popit
            ; Pop up a level after the following statements are executed.
      (progn
        (addr new-regs ss-reg
          (build-sentences-frame (getr new-regs s-reg)))
            ; Build a sentences frame and place it in a register.
        (setq good-grammar
          (append good-grammar (getr new-regs ss-reg)))
            ; Add the new frame to a global.
        (remr new-regs s-reg))))
            ; Remove the register containing the frame created by the sent ATN.
```

As a sentence is parsed its parts - verb, direct object, and prepositional phrases - are identified and placed into sentence structure frames (figure 4). A verb is the basic requirement for a sentence; a direct object is not always necessary and neither are prepositional phrases. When a verb is precessed by the parser, it's sense is looked up in a dictionary and placed in the verb frame. Verb senses allow different verbs with similar meanings to be grouped under one term. Within the direct object phrase, there is an optional determiner, one or more optional adjectives, and one or more nouns. The only conjunction for connecting adjectives or nouns is 'and.' Multiple sentences may be entered at once as long as they are separated by periods. Figure 5 shows some legal and illegal sentences that can be entered and figure 6 shows a sentence placed in sentence structure frames.

BEST COPY AVAILABLE

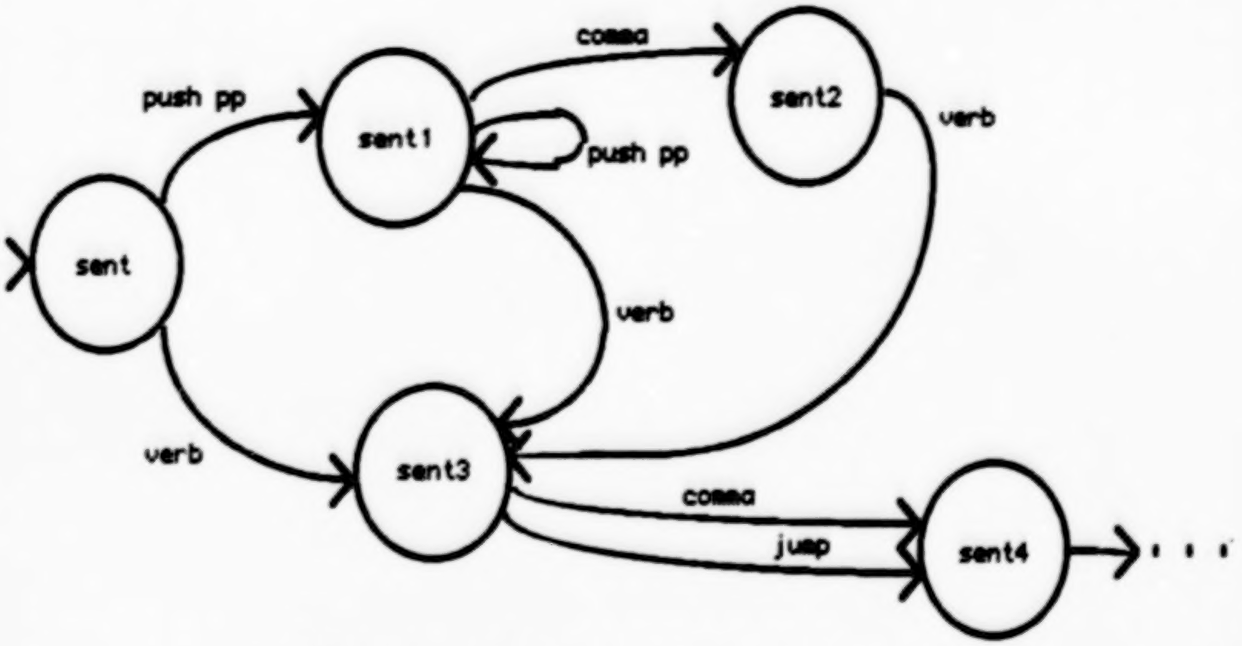


Figure 3. Sentence ATN (partial)

The nodes for all ATNs are ordered so that the path most likely to be successful is tried first. In the setences ATN, the verb transition is checked before a preposition transition. Transitions to next states are placed on a LIFO stack which is later popped to determine the next node to execute. So the information about the "push pp" transition from node sent is placed on the stack before information about the "verb" transition. When the stack is popped the "verb" transition is on top and tried first.

The knowledge about words is kept in a dictionary which lists the words allowed by the MJI and the parts of speech they may fill. Figure 7 contains some dictionary entries. The dictionary contains two special entries - *number and *char-string. *number is used to allow numbers within sentences. Whenever a number is specified in a sentence, the *number entry is used for data dictionary lookups. The other special entry is *char-string. Any time a word cannot be found in the dictionary and the word is not a number, the *char-string entry is used for the word. This feature is necessary to allow for words which cannot be entered into the dictionary ahead of time - such as the name of a newly created wildcard page or STOL procedure.

Because many words in the English language can fill the role of more than one part of speech, it is possible for a single sentence to have more than one grammatically correct interpretation. In the MJI, the parser outputs a single grammatically correct interpretation of a sentence. If the interpretation produced cannot be changed into directives, another grammatically correct interpretation is produced (if one exists). If a sentence is ambiguous, the first parsing produced is not necessarily the correct parsing.

BEST COPY AVAILABLE

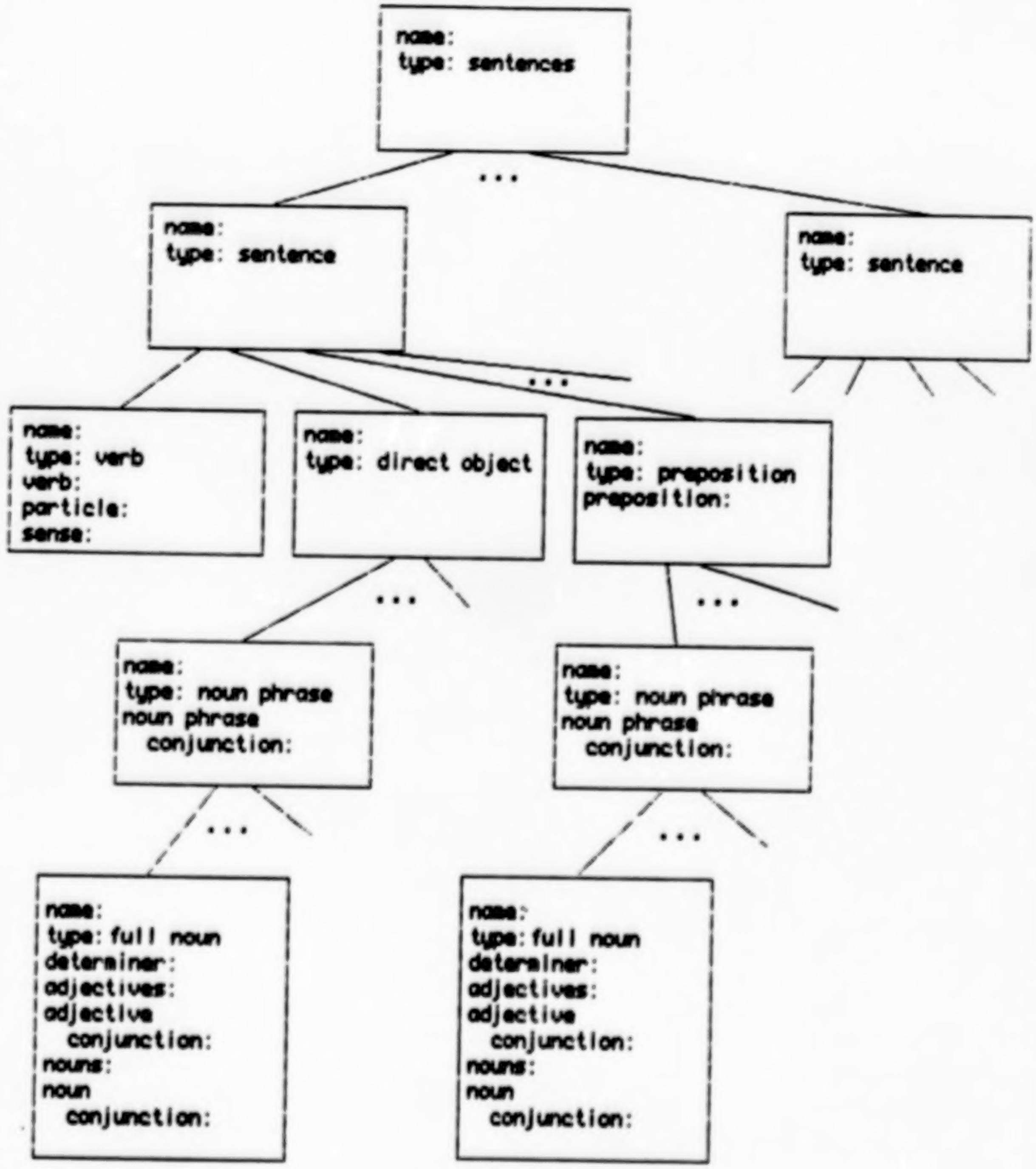
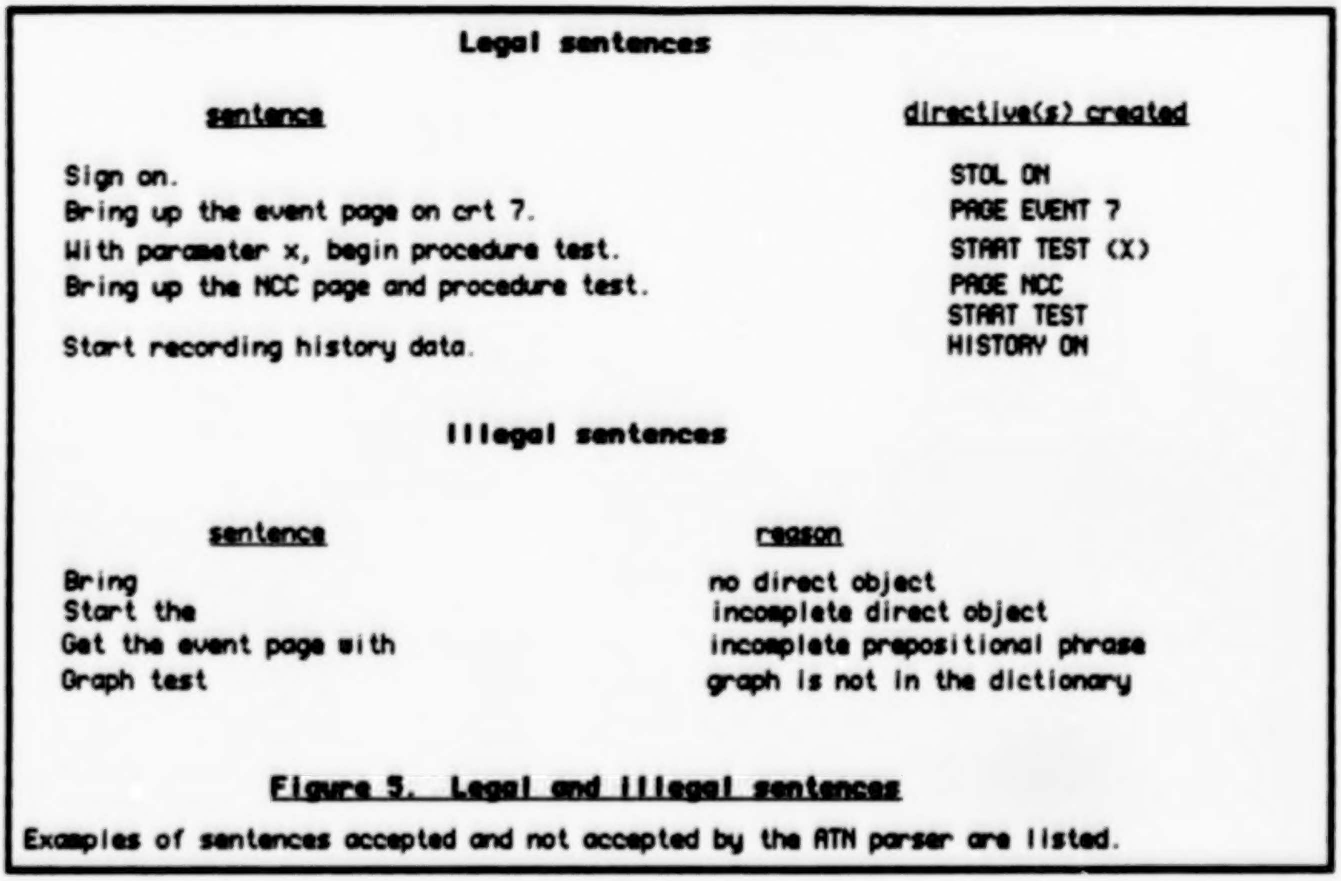

name:
type: sentences
...
name:
type: sentence
name:
type: sentence
...
name:
type: verb
verb:
particle:
sense:
name:
type: direct object
name:
type: preposition
preposition:
...
...
name:
type: noun phrase
noun phrase
conjunction:
name:
type: noun phrase
noun phrase
conjunction:
...
...
name:
type: full noun
determiner:
adjectives:
adjective
conjunction:
nouns:
noun
conjunction:
name:
type: full noun
determiner:
adjectives:
adjective
conjunction:
nouns:
noun
conjunction:

Figure 4. Blank sentence structure

As input is processed by the ATN parser it is placed in sentence structure frames.

BEST COPY AVAILABLE



Legal sentences

sentence	directive(s) created
Sign on.	STOL ON
Bring up the event page on crt 7.	PAGE EVENT 7
With parameter x, begin procedure test.	START TEST (X)
Bring up the NCC page and procedure test.	PAGE NCC START TEST
Start recording history data.	HISTORY ON

Illegal sentences

sentence	reason
Bring	no direct object
Start the	incomplete direct object
Get the event page with	incomplete prepositional phrase
Graph test	graph is not in the dictionary

Figure 5. Legal and Illegal sentences

Examples of sentences accepted and not accepted by the ATN parser are listed.

The rule-based system

Once a sentence is successfully parsed and placed in sentence structure frames, it's meaning is determined by submitting the frames to a rule-based system (figure 8). First pertinent information from the frames, such as the verb sense and direct object adjectives and nouns, is moved into a working knowledge base of fact name and value pairs (figure 9).

The inference engine is forward-chaining and does a depth first search for a solution. It examines rules to see if any can be fired. When the conditions in the antecedents of rules are met by the current set of facts in the working knowledge base, the actions specified in the consequents of the rules are executed. The consequents are used to add facts to or modify facts in the working knowledge base. The consequents may also specify procedures which are called for their side effects of modifying the working knowledge base or printing results for the user. The inference engine keeps examining rules until a solution is found or no more rules can be fired. Figure 10 contains some rules.

The rules are split into several contexts each containing rules for specific functions. For example, the context PAGE-CHECK contains the rule page-check-1 and page-check-2 for determining if the direct object is a page and MAKE-START-DIR contains the rules make-

BEST COPY AVAILABLE

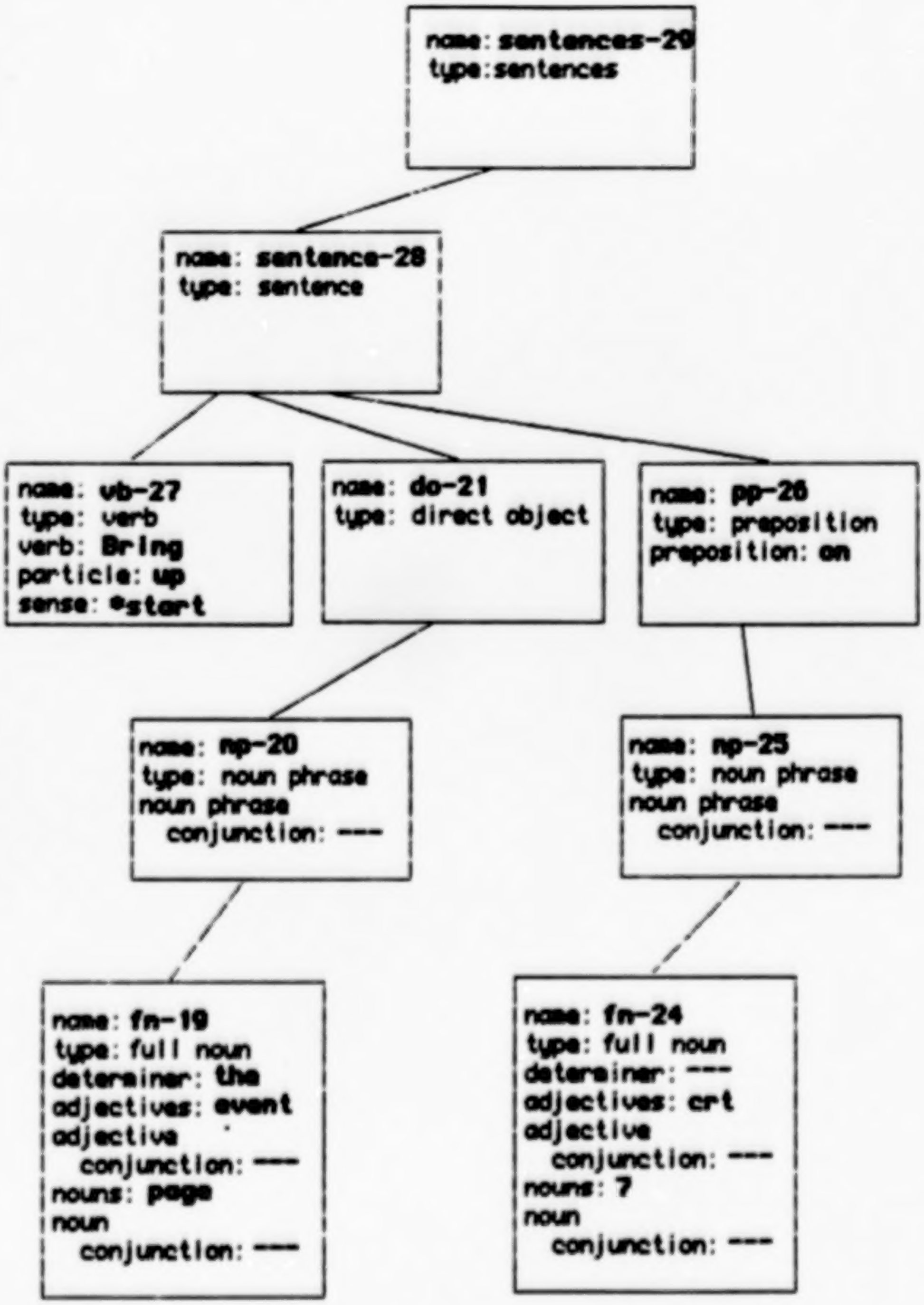

name: sentences-29
type: sentences
name: sentence-28
type: sentence
name: vb-27
type: verb
verb: Bring
particle: up
sense: *start
name: do-21
type: direct object
name: pp-26
type: preposition
preposition: on
name: np-20
type: noun phrase
noun phrase
conjunction: ---
name: np-25
type: noun phrase
noun phrase
conjunction: ---
name: fn-19
type: full noun
determiner: the
adjectives: event
adjective
conjunction: ---
nouns: page
noun
conjunction: ---
name: fn-24
type: full noun
determiner: ---
adjectives: crt
adjective
conjunction: ---
nouns: 7
noun
conjunction: ---

Figure 6. Filled sentence structure frames
The sentence "Bring up the event page on crt 7" is placed into sentence structure frames as it is parsed.

BEST COPY AVAILABLE

word	category	value(s)
begin	word-class	verb
begin	sense	*start
clear	word-class	verb
clear	sense	*clear
crt	word-class	noun, adjective
crt	number	singular
crts	word-class	noun, adjective
crts	number	plural
*char-string	word-class	noun, adjective
*char-string	number	singular, plural
*number	word-class	noun, adjective
*number	number	singular, plural

Figure 7. Dictionary entries

Each word in the dictionary has a word class entry indicating the part of speech the word may fill. Verbs also have a sense entry and nouns and adjectives have a number entry indicating whether the word is singular or plural.

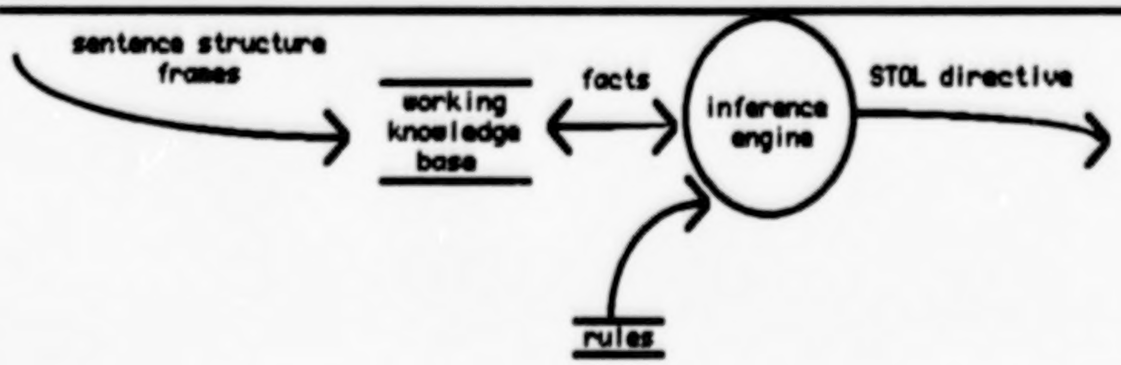


Figure 8. Rule-based system

Information from sentence structure frames is moved into the inference engine's working knowledge base. The engine then uses the working knowledge base to determine which rules should be fired to produce STOL directives.

start-dir-1 and make-start-dir-2 for making START directives. One of the facts in the working knowledge base tells the inference engine which rule context to use.

BEST COPY AVAILABLE

Working knowledge base

fact name	fact value
SENSE	*START
ADJS	EVENT
NOUN	PAGE
PREP-PHRASES	#:|pp-26|
CURRENT-CONTEXT	WHICH-DO-CHECK1

Figure 9. Working knowledge base

The working knowledge base contains fact-value pairs. The facts making up the initial working knowledge base for the sentence "Bring up the event page on crt 7" are shown. This information is used to determine which rules may be fired (when the data in the working knowledge base make rules' antecedents true) by executing the rules' consequents (which may cause changes to the working knowledge base).

The explanation system

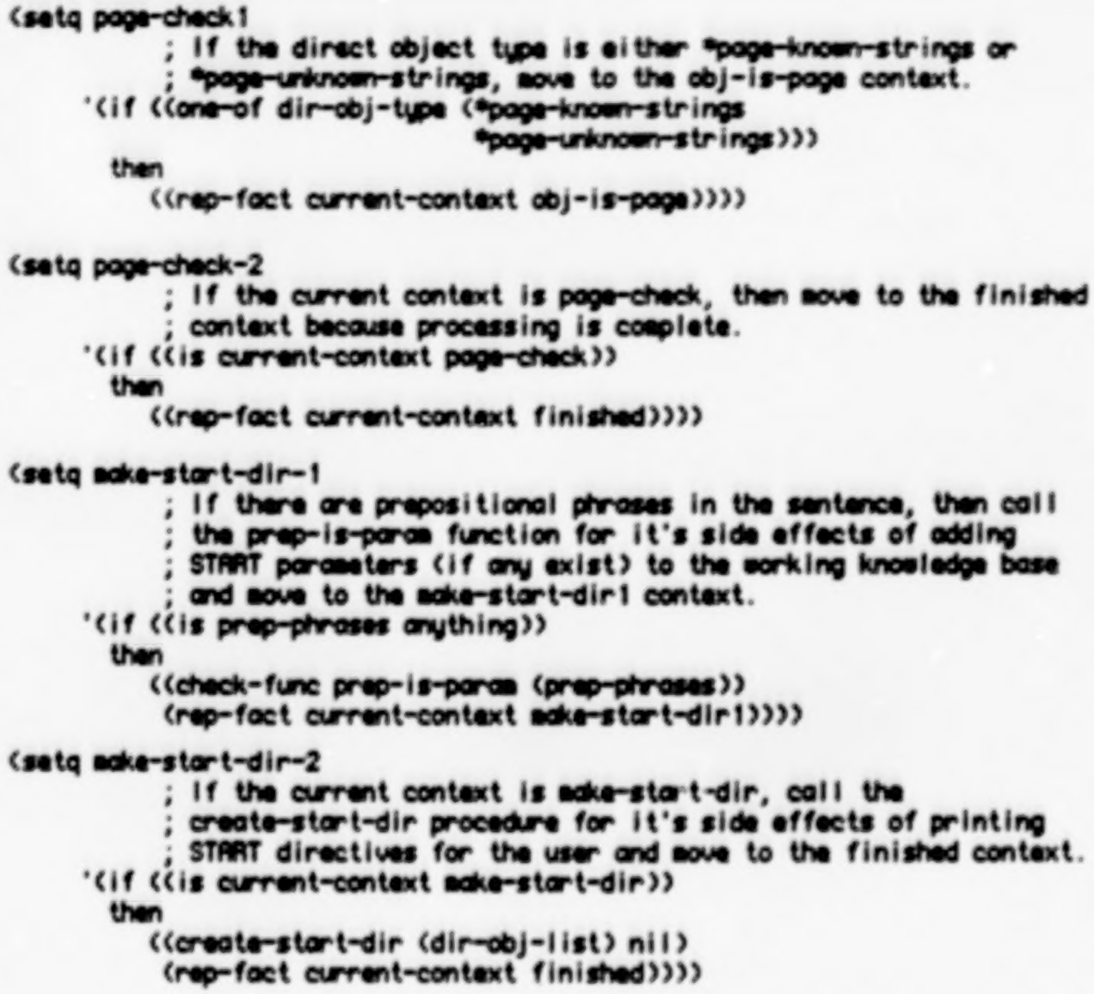
The explanation system allows the user to see the steps the MJI went through to translate a sentence into STOL directives. After an interpretation is produced, the user may enter 'explain.' The MJI will then describe how the sentence was grammatically typed and list the rules which were fired to obtain the directives. The rules are printed in natural language format with substitutions made from saved copies of the working knowledge base for values in antecedents and consequents. Only the rules which directly lead to a solution are described. Figure 11 shows the explanation of the sentence 'Bring up the event page on crt 7.'

In order to give an explanation, the MJI saves the sentence structure frame of the last sentence processed. The MJI also saves the names of the rules fired on the path to a solution along with copies of the working knowledge base at the time the rules' antecedents were checked.

Conclusions

The MSOCC Jargon Interpreter demonstrates the successful translation of English sentences into STOL directives. The MJI operates in a constrained domain consisting of clearly defined objects (crts, display pages, STOL procedures, and history files) and definite operations upon each of the objects specifiable by the sentences' verbs. Additions to MJI's rules and dictionary can expand the capacity of the interpreter allowing for more directives.

BEST COPY AVAILABLE



```
(setq page-check1
        ; If the direct object type is either *page-known-strings or
        ; *page-unknown-strings, move to the obj-is-page context.
    '(if ((one-of dir-obj-type (*page-known-strings
                                *page-unknown-strings)))
      then
       ((rep-fact current-context obj-is-page))))

(setq page-check-2
        ; If the current context is page-check, then move to the finished
        ; context because processing is complete.
    '(if ((is current-context page-check))
      then
       ((rep-fact current-context finished))))

(setq make-start-dir-1
        ; If there are prepositional phrases in the sentence, then call
        ; the prep-is-param function for it's side effects of adding
        ; START parameters (if any exist) to the working knowledge base
        ; and move to the make-start-dir1 context.
    '(if ((is prep-phrases anything))
      then
       ((check-func prep-is-param (prep-phrases))
        (rep-fact current-context make-start-dir1))))

(setq make-start-dir-2
        ; If the current context is make-start-dir, call the
        ; create-start-dir procedure for it's side effects of printing
        ; START directives for the user and move to the finished context.
    '(if ((is current-context make-start-dir))
      then
       ((create-start-dir (dir-obj-list) nil)
        (rep-fact current-context finished))))
```

Figure 10. Rules

Shown are some of the rules used by the MJI.

The combination of ATN parser and rule-based system can be used in similarly constrained domains. For example, the rules and dictionary of the MJI are being modified so it can be used as a front-end for a display creation prototype under development. The new prototype is a customized graphics editor which allows the creation of wildcard display pages for use in MSOCC.

BEST COPY AVAILABLE

```
Your original sentence was grammatically typed as follows.
  Verb
    verb:  BRING  particle:  UP  verb sense:  *START
  Direct ojbect
    determiner:  (THE)  adjective:  (EVENT)  nouns:  (PAGE)
  Prepositional phrase
    preposition:  ON  adjectives:  (CRT)  nouns:  (7)
Your original input was sent directly through the inference engine for translation
to STOL directives.
                                    ____________
The current rule is WHICH-DO-CHECK1-2.
Since the value of variable CURRENT CONTEXT is WHICH-DO-CHECK1.
The old value of CURRENT-CONTEXT was replaced with WHICH-DO-CHECK2.
                                    ____________
The current rule is WHICH-DO-CHECK2-1.
Since the value of variable SENSE is *START and is one of *CLEAR *CLOSE *DISPLAY
  *INIT *OPEN *PUT *RESET *START *STOP.
Since the value of variable NOUN is PAGE and is not null.
A function IS-PAGE was called with parameters
   ADJS:  EVENT
   NOUN:  PAGE
 to deduce the fact(s)
   DIR-OBJ-TYPE with value *PAGE-KNOWN-STRINGS
   DIR-OBJ-LIST with value EVENT.
The old value of CURRENT-CONTEXT was replaced with PAGE-CHECK.
                                    ____________
The current rule is PAGE-CHECK-1.
Since the value of variable DIR-OBJ-TYPE is *PAGE-KNOWN-STRINGS and is one of
  *PAGE-KNOWN-STRINGS *PAGE-UNKNOWN-STRINGS.
The old value of CURRENT-CONTEXT was replaced with OBJ-IS-PAGE.
                                    ____________
The current rule is OBJ-IS-PAGE-2.
Since the value of variable DIR-OBJ-TYPE is *PAGE-KNOWN-STRINGS.
Since the value of variable SENSE is *START and is one of *DISPLAY *INIT *OPEN
  *PUT *START.
The old value of CURRENT-CONTEXT was replaced with MAKE-PAGE-DIR.
```

(continued on next page)

Figure 11. Explanation example

When giving an explanation, the NLI first tells how the sentence was typed and then lists the rules executed to produce STOL directives. Substitutions from saved copies of the working knowledge base are made into the rules before they are displayed.

Acknowledgments

I would like to thank Henry Murray of NASA Goddard's Control Center Systems Branch and Mike Blackstone of Unisys for their support of the work described in this paper. I would also like to thank them and Joe Maynard of Unisys for reviewing this paper.

BEST COPY AVAILABLE

```
        The current context is MAKE-PAGE-DIR-1.
        Since the value of variable PREP-PHRASES is #:|pp-26| and is not null.
        A function PREP-IS-CRT was called with parameters
           PREP-PHRASES:  #:|pp-26|
          to deduce the fact(s)
           PREP-OBJ-CRT with value *CRT-KNOWN-NUMBERS
           PREP-CRT-LIST with value 7.
        A function PREP-IS-INTERVAL was called with parameters
           PREP-PHRASES:  #:|pp-10|
          but did not deduce any new facts.
        The old value of CURRENT-CONTEXT was replaced with MAKE-PAGE-DIR1.
                                          ____________

The current rule is MAKE-PAGE-DIR1-1.
Since the value of variable CURRENT-CONTEXT is MAKE-PAGE-DIR1.
A function CREATE-PAGE-DIR was called with parameters
    UP-PAGE
    DIR-OBJ-LIST:  EVENT
    PREP-CRT-LIST:  7
    PREP-INT-LIST:
  which created the STOL directive(s)
    PAGE EVENT 7
The old value of CURRENT-CONTEXT was replaced with FINISHED.
                                          ____________

                              EXPLANATION COMPLETE
```

Figure 11. Explanation example (continued)

When giving an explanation, the MJI first tells how the sentence was typed and then lists the rules executed to produce STOL directives. Substitutions from saved copies of the working knowledge base are made into the rules before they are displayed.

References

Allen, J., Natural Language Understanding, Benjamin/Cummings, California, 1987.

Denning, P. J., J. B. Dennis and J. E. Qualitz, Machines, Languages, and Computation, Prentice-Hall, New Jersey, 1978.

UNISYS, Applicability Of Artificial Intelligence In The Multi-Satellite Control Center (MOSCC) Software Design Document, August 1988.

UNISYS, CCS-7SUG/0485, MSOCC Applications Executive (MAE) System Test And Operations Language (STOL) Programmers Guide, September, 1987.

Winograd, T., Language As A Cognitive Process Volume I: Syntax, Addison-Wesley, Massachusetts, 1983.

Winston, P. H., Artificial Intelligence, Addison-Wesley, Massachusetts, 1984.

BEST COPY AVAILABLE

Planning and Scheduling

BLANK PAGE

Ground Data Systems Resource Allocation Process

Carol A. Berner, Ralph Durham and Norman B. Reilly
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

Abstract

The Ground Data Systems Resource Allocation Process at the Jet Propulsion Laboratory (JPL) provides medium- and long-range planning for the use of Deep Space Network and Mission Control and Computing Center resources in support of NASA's deep space missions and Earth-based science. Resources consist of radio antenna complexes located in California, Australia and Spain, and associated data processing and control computer networks at JPL. Until 1985, JPL used a manual planning process that was extremely labor intensive and provided a planning horizon of only two to three weeks. With approval of increasingly complex missions, a more efficient planning system was needed to optimize the use of supporting facilities, minimize contention among users, expand the planning horizon to several years and reduce manual labor.

To support an improved planning methodology, a semi-automated system has been developed that not only achieves these objectives but also enhances scientific data return. This end-to-end system allows operations personnel to interactively generate, edit and revise allocation plans spanning periods of up to ten years based on the relative merit of mission events.

*An integral part of this system is a software system known as the **R**esource **A**llocation and **P**lanning **H**elper (RALPH). RALPH merges the conventional methods of operations research, rule-based knowledge engineering and advanced data base structures. RALPH employs a generic, highly modular architecture capable of solving a wide variety of scheduling and resource sequencing problems. Because RALPH easily handles generic requirements, the system has had an important influence on the simplification and standardization of input requirements from all projects. Additionally, the system design provides adaptability to changing mission environments.*

The rule-based RALPH system has saved significant labor in the resource allocation process at JPL. Its successful use affirms the importance of establishing and applying event priorities based on scientific merit, and the benefit of continuity in planning provided by knowledge-based engineering. The RALPH system exhibits a strong potential for minimizing development cycles of resource and payload planning systems throughout NASA and the private sector.

Resource Allocation Planning at the Jet Propulsion Laboratory

The Ground Data Systems Resource Allocation Process was established at the Jet Propulsion Laboratory (JPL) for the purpose of efficiently managing the use of ground data system resources. These resources include the Deep Space Network (DSN) and the Mission Control and Computing Center (MCCC).

The DSN is a global network of three tracking and data communications complexes which support NASA interplanetary missions and Earth-based science. Each complex has a 70 meter (diameter) antenna, two 34 meter antennas, and a 26 meter antenna. The four antennas are linked to a Signal Processing Center computer network at each complex. Each of the three Signal Processing Centers is linked by high-speed and wideband circuits to the MCCC data systems residing at JPL.

The MCCC is a centralized multimission data processing and control center. It has the resources required for the successful conduct and control of all aspects of space flight missions and radio-based astronomical observations. These include performing real-time multimission operations, data processing and systems control, and non-real time computations, such as image processing and data records generation.

The JPL Flight Project Support Office (FPSO) is responsible for allocating DSN and MCCC ground data system resources for spacecraft tracking, communications and science activities. The output of the resource allocation process is plans for optimal use of the available system resources. These plans must support the requirements of all missions. As a minimum, survival support must be guaranteed to spacecraft whose missions have been granted extensions beyond their primary objectives.

Understanding Planning Constraints

Allocation plans are required to address the needs of all projects, each of which may be in different phases of their lifecycles: missions in flight, in pre-flight development or in advanced planning stages. Allocation plans must be based on a set of constraints consisting of viewperiods, project requests, and system resource requirements. Each of these constraints is affected by dynamic factors that influence the planning strategy from week to week, complicating the planning problem.

A viewperiod is a time interval over which a spacecraft is in view of a particular station antenna, allowing tracking and communications with the spacecraft. The ground stations are spaced at intervals of about 120 degrees in longitude from one another, ensuring that a spacecraft will be in view of at least one ground station at any time as the Earth rotates. Thus, each antenna at a single complex will have a viewperiod for each spacecraft target at least once during each Earth rotation. The dynamics of each spacecraft flight path determine the exact time and duration of every viewperiod.

Project requests come in two different forms. There are specific requests and generic requests. Specific requests cover critical, time-dependent events. Projects typically demand specific tracking coverage to compensate for restrictions imposed by viewperiods or flight paths. A planetary encounter is one such time-dependent event. This level of input severely reduces planning flexibility.

Generic user requests tend to be non-time dependent. A project user will ask for a total amount of support and a broad class of equipment over an extended time period. An example is the minimum antenna coverage necessary to ensure signal reception. Combinations of antennas may be required to pull in very distant signals. Minimum and maximum allowable activity separations is another typical requirement. At the most general level of input, a user requests the fulfillment of project objectives within a specific time frame. Such generic requests afford the most latitude in planning equipment allocations for a particular window of time.

System requirements place another set of constraints on the resource allocation process. Certain station equipment must undergo pre- and post-calibration. Station crews typically require a minimum of half an hour separation between successive signal acquisitions to make these necessary configuration changes. Resource maintenance, downtime and unmanned periods also contribute to the complexity of the planning problem.

The Resource Allocation Problem

From the inception of the Deep Space Network in 1958 through early 1986, resource allocation was essentially a manual process. Even for typical tracking situations, planning is a complex problem with many variables. The high level of complexity dictates that the solution be determined in a somewhat non-systematic way that usually involves a great deal of human decision. In a joint effort with the DSN, MCCC, Flight Projects, and other users, FPSO established several teams to accomplish the goals of the resource allocation process.

The Joint Users Resource Allocation Planning Committee (JURAPC) includes the managers and scientists of all involved organizations. It functions to establish allocation policies and priorities and to review proposed support requirements, established plans, and requests to modify requirements.

The Resource Allocation Planning Team (RAPT), a subcommittee of the JURAPC, is chartered to implement established policies, gather support requirements, review resource allocation plans, establish event priorities and resolve conflicts through negotiations. The RAPT is the focal point for all contention studies and "what-if" feasibility studies.

The Resource Analysis Team (RAT) consists of expert planners and analysts. The workhorse of the resource allocation process, it operates and maintains the tools necessary for production of plans and special studies, and for database administration. The RAT team leader also coordinates and mediates all RAPT meetings.

Using the manual planning system, this labor-intensive approach to resource allocation was capable of providing a planning horizon of only two to three weeks. The lack of a structured method for determining feasible support led to the generation of allocation plans which did not adequately reflect user needs. As a result, many hours of negotiations per week were necessary in order to clear unforseen conflicts.

Over the years, the number of approved missions has been growing and their data collection and transmission capabilities have been expanding. Compounding the problem, most of the DSN-supported spacecraft have proceeded in the same general directions away from Earth, causing viewperiods to overlap considerably. Because of this overlap, the concentration of requests for resources has caused an uneven allocation profile. It became imperative to develop a more efficient planning system to optimize the use of supporting facilities, minimize contention between users, expand the planning horizon to several years, and reduce manual labor.

Development of a Software Tool

JPL established a Design Team to develop a tool that gives the Resource Analysis Team the ability to achieve a more efficient planning system. This Design Team joined planning analysts and software developers in an environment that enhanced information exchange (Figure 1). This integration of operations and development provided a global view of the task. Operational goals and developmental capabilities were merged to create a software tool that assists in all facets of the resource allocation process. It was named the **R**esource **AL**location **P**lanning **H**elper, or RALPH. The relationship of RALPH to the resource allocation process is illustrated in Figure 2.

The Design Team established clear top-level goals for the RALPH system:

- Generation of detailed resource allocation plans
- Reduction of plan generation time
- Reduction of user conflict negotiation time
- Quick turn-around for special studies
- Automation of resource selection
- Allocation based on event prioritization

Important new concepts were introduced to optimize resource planning:

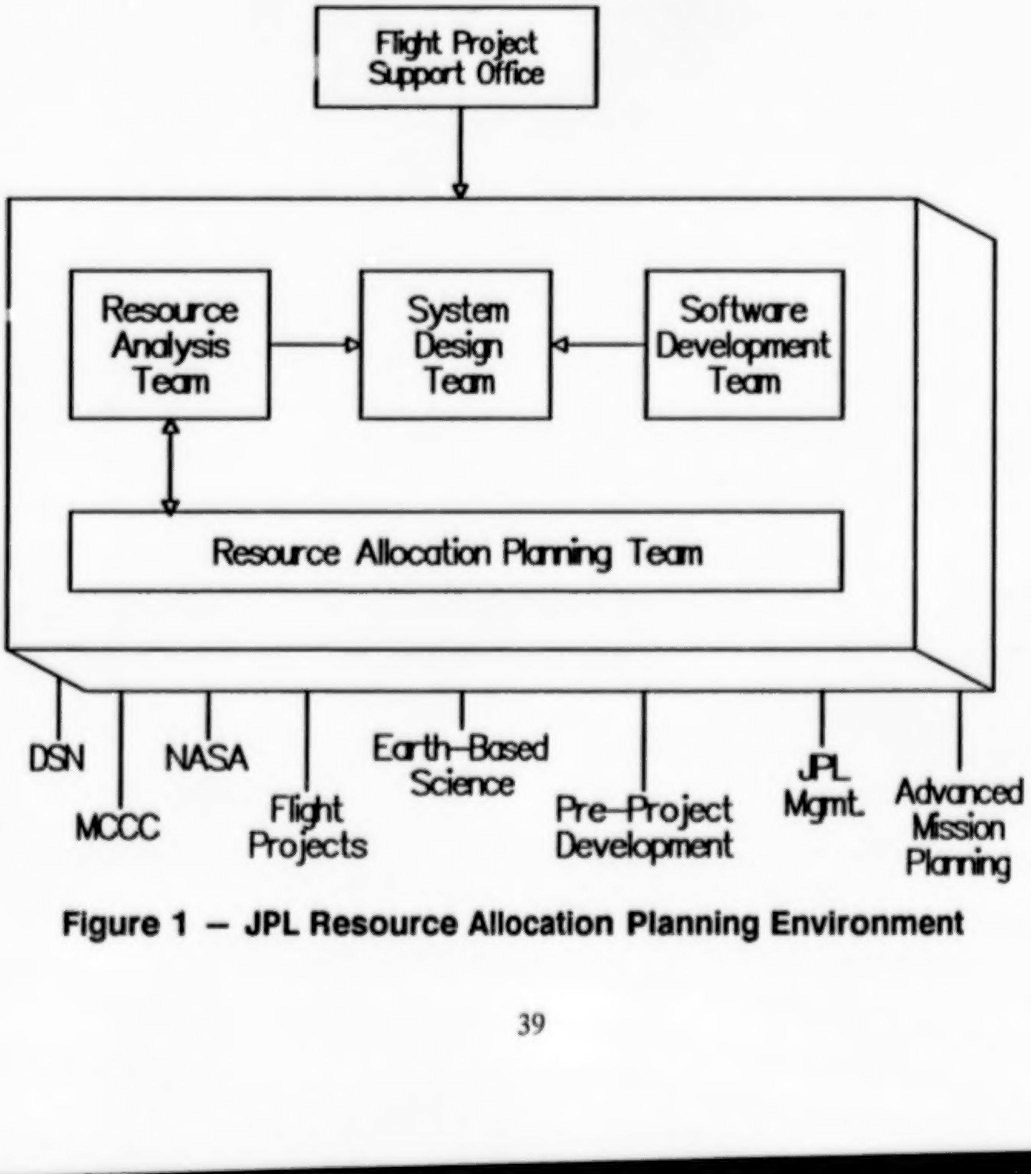

Flight Project Support Office
Resource Analysis Team
System Design Team
Software Development Team
Resource Allocation Planning Team
DSN
MCCC
NASA
Flight Projects
Earth-Based Science
Pre-Project Development
JPL Mgmt.
Advanced Mission Planning

Figure 1 — JPL Resource Allocation Planning Environment

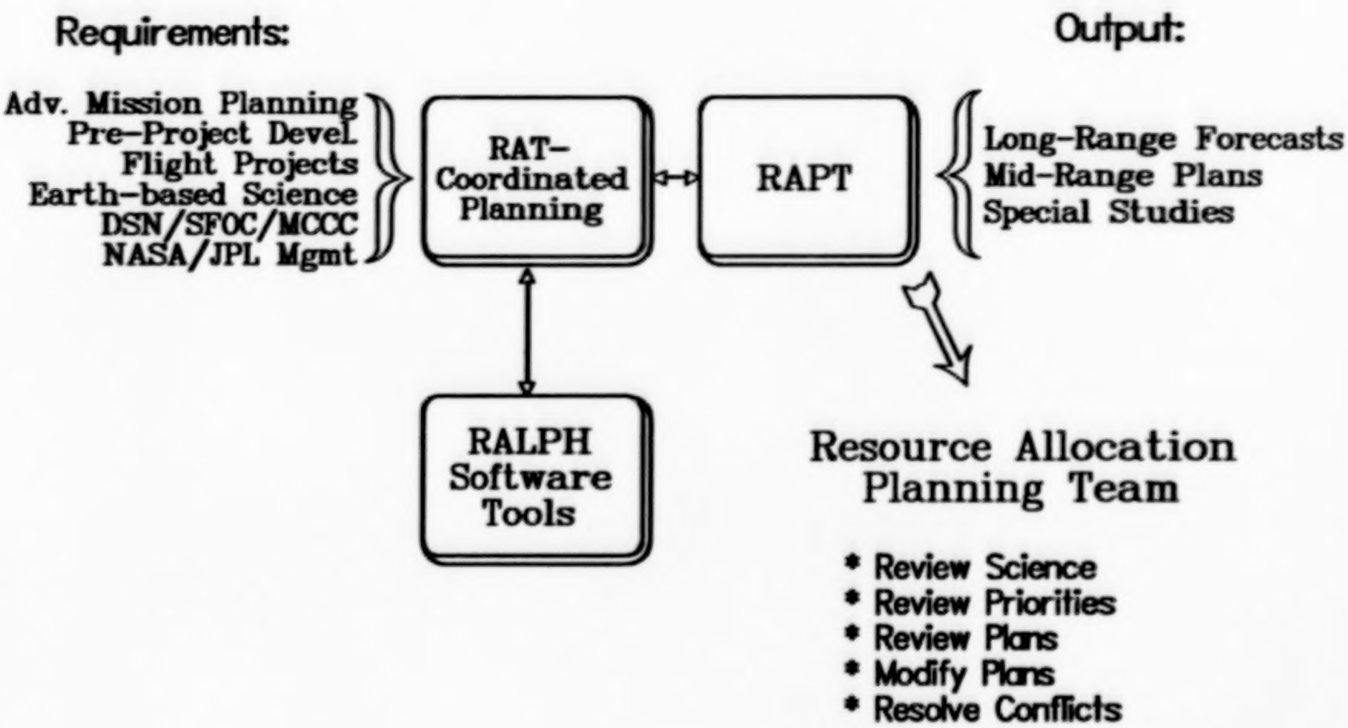

Requirements:
Output:
Adv. Mission Planning
Pre-Project Devel.
Flight Projects
Earth-based Science
DSN/SFOC/MCCC
NASA/JPL Mgmt
RAT-Coordinated Planning
RAPT
Long-Range Forecasts
Mid-Range Plans
Special Studies
RALPH Software Tools
Resource Allocation Planning Team
* Review Science
* Review Priorities
* Review Plans
* Modify Plans
* Resolve Conflicts

Figure 2 — JPL Mechanism of Resource Allocation

- Maximization of science return, followed by optimization of resource utilization
- Identification of the variance between resource allocation plans and the sum of the original requirements (e.g., NASA Support Plans)

Implementation of these concepts would enable the planning analysts to better manage the resource allocation process and improve internal flight project planning and scheduling processes. Previous allocation techniques have been based on mission priority. Planning analysts at JPL have pioneered the idea of levels of support based on scientific merit or "event" priority. This method gives project negotiators a clear understanding of the impact on science return caused by any compromises to the plan.

It is a goal of the resource allocation process to create 100% conflict-free resource allocation plans. However, oversubscription of resources, as well as the viewperiod overlap problem, creates a situation such that requests can not be completely satisfied. RALPH allows the resource analysis team to provide project users with near-optimum resource allocation plans which meet all user requirements while clearly pinpointing any remaining conflicts. It remains a human decision-making task for project users to jointly negotiate a final determination of acceptable conflict-free levels of support. The final product is then used for mission operations schedules.

Accommodating the Project Lifecycle

The resource allocation process was designed to accommodate project needs throughout the various phases of their lifecycles. For this reason the resource allocation process provides products for three distinct subprocesses:

- Special studies
- Long-range forecasting
- Mid-range planning

Special studies are generated throughout the project lifecycle. Approximately ten years before a proposed spacecraft launch, when a new project defines preliminary functional requirements for spacecraft design and equipment support, the Resource Analysis Team reviews those requirements in light of approved resource allocation plans and major events. These special studies provide valuable impact assessments early in the project lifecycle.

At any time, projects may request special studies to assess the feasibilities of different planning options, such as the impact of moving spacecraft launch windows. If contention appears inevitable, it is identified within the report, along with appropriate statistical analyses and recommendations for alleviating the conflict.

Between two to ten years before a project's launch, the feasibility of requirements is seriously considered. This is the first point at which all requirements, such as the Support Instrumentation Requirements Document, the NASA Support Plan and the Mission Support Plan, are brought together and considered in total. At this stage, the requirements contain sufficient detail to generate a resource allocation forecast. However, the information during this period is tentative and subject to change based on many different factors. Despite their tentative nature, these long-range forecasts provide valuable resource allocation overviews during the mission's planning phase. Forecasts assist in the process of enhancing science return and reducing user impact. In addition, they supply information that is instrumental to advanced planning and resource loading studies.

Mid-range planning occurs between eight weeks to two years before an event. The resource allocation phase of the project lifecycle occurs from two years to six months prior to an activity. During this period, resource allocation plans are provided to project users for mission sequence development. This period marks the beginning of the iterative human decision-making process of conflict resolution between all project users, conducted by the Resource Analysis Team in RAPT and JURAPC meetings.

Between six months and eight weeks before a planned activity, all conflicts must be resolved. This period supports the operations phase of the project lifecycle. At eight weeks prior to launch, the DSN and MCCC are presented with detailed conflict-free resource allocation plans for real-time operations. Figure 3 is an overview of the resource allocation process.

Certain products were identified to support each phase of the allocation process. These were selected as output candidates for the RALPH system design.

Special Studies
- Impact studies
- Management overviews

Long-Range Forecasting
- Detailed long-range forecasts
- Summaries and supporting analyses

Mid-Range Planning
- Detailed resource allocation plans
- Conflict summaries
- Resource loading summaries

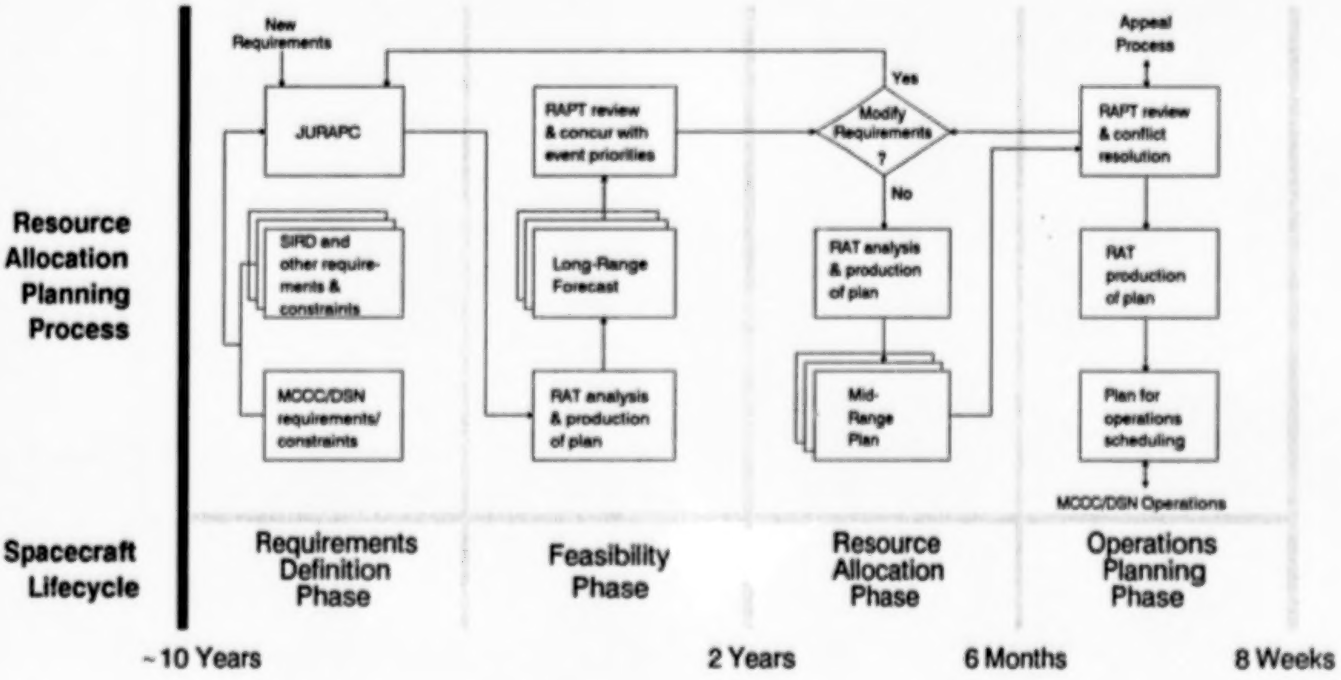

New Requirements
JURAPC
SIRD and other requirements & constraints
MCCC/DSN requirements/ constraints
RAPT review & concur with event priorities
Long-Range Forecast
RAT analysis & production of plan
Yes
Modify Requirements ?
No
RAT analysis & production of plan
Mid-Range Plan
Appeal Process
RAPT review & conflict resolution
RAT production of plan
Plan for operations scheduling
MCCC/DSN Operations
Resource Allocation Planning Process
Spacecraft Lifecycle
Requirements Definition Phase
Feasibility Phase
Resource Allocation Phase
Operations Planning Phase
~10 Years
2 Years
6 Months
8 Weeks

Figure 3 — JPL Ground Data System Resource Allocation Process

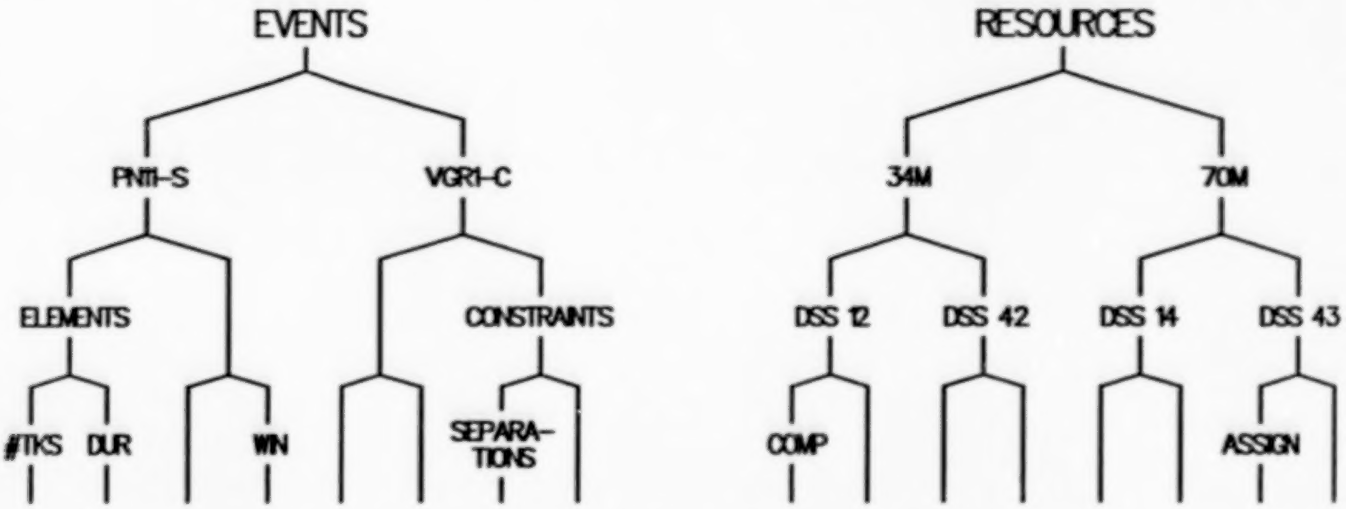


Figure 4 – Hierarchical Tree Structure of Data Representation

With top-level goals and products clearly defined, the next step was to model the processes of the resource allocation problem.

Modeling Resource Allocation

RALPH is implemented as an expert system that utilizes hierarchical tree structures. These tree structures contain descriptions of resources, requirements and specific rules. Data within these structures are easily modified to represent new conditions (Figure 4).

The formal analytical processing capabilities of Operations Research (OR) have been included in modules that solve problems of resource allocation. Other modules include rules for attribute assignment, constraint violation, event language entry, and syntax parsing of user support events. All of these modules form a generic section of code, called the Core Resource Allocation Modules (CRAM). Each CRAM module is designed to utilize the symbolic logic and predicate calculus of the tree structures. In this way, the synthesis of knowledge engineering and OR technologies has proven to be a successful solution to the JPL resource allocation problem.

The entire resource allocation problem can be decomposed into five functional steps. This five-rule set, initially proposed by Information Sciences, Inc. in the 1970's, provides a paradigm of the way a human analyst solves an allocation problem. The steps are:

1. Definition of user support goals and feasible support descriptions
2. Prioritization of feasible support
3. Resource assignment
4. Time assignment
5. Plan verification and conflict resolution

The five-rule set suggested the need for several software subsystems to support the resource allocation planning function. These subsystems, a requirements translator, a mid-range plan generator, a long-range forecast generator, text and graphics editors, and an output capability generator are linked to provide a linear data flow through the system (Figure 5).

The requirements translator is a high-level language processor. User event support descriptions are entered through language syntax processing modules which prompt the operator for appropriate attribute entries. The event description syntax is stored as a string. User support events may have attributes such as duration of assignments, resource preferences, and temporal relationships.

Priorities are assigned based on the scientific significance of the user events. This concept was originated by the Resource Allocation Planning Team as the best means of achieving maximum science return with limited data system resources. The priority system, depicted in Table 1, is applied to all users for planning purposes. In this numerical scheme, an assignment of "one" represents the highest priority.

The plan and forecast generator subsystems parse syntax strings into standard event tree structures. The generator must then allocate resources and time to those events that are consistent with the resource assignment objects contained within the system under the following guidelines:

- Fully satisfy each user event
- Minimize conflict among users
- Maximize resource utilization

To reduce the combinatorial complexity that is typically encountered in large-scale resource scheduling systems, the generator uses a two-pass approach. The first pass determines probabilistic profiles of resource usage determined by requirements, windows and hard constraints. The second pass uses these profiles to evaluate possible scheduling points for minimum conflict. It proceeds in order of event priority, transforming the probabilistic profiles into deterministic schedules.

Finally, RALPH employs two types of resource assignment editors for plan verification and conflict resolution. Change entries are verified by many of the same CRAM attribute validation modules used by the requirements translator for event object descriptions.

When validating and enhancing allocation plans, planning analysts may use the RALPH graphics editor. This allows easy modification of the plan based on operator recognition of color-coded visual allocation patterns (Figure 6).

Modifications may also be entered through the RALPH text editor. This editor is designed to mimic the manual paper process used in RAPT negotiation meetings. It consists of line listings of resource allocations sorted by day of week, time and user. The paper-based process will soon be eliminated entirely in favor of computer-aided editing.

Results

The RALPH software is implemented on a DEC MicroVAX II customized with a GPX high-resolution graphics workstation. RALPH is fully operational, allowing the Resource Analysis Team to efficiently produce long-range forecasts and mid-range plans for the DSN and MCCC. The number of workhours required to produce a one-week mid-range plan have been reduced from 25 hours of manual labor to five hours using the RALPH tool. Without RALPH, long-range forecasts were not possible. With RALPH, long-range forecasts have been enabled.

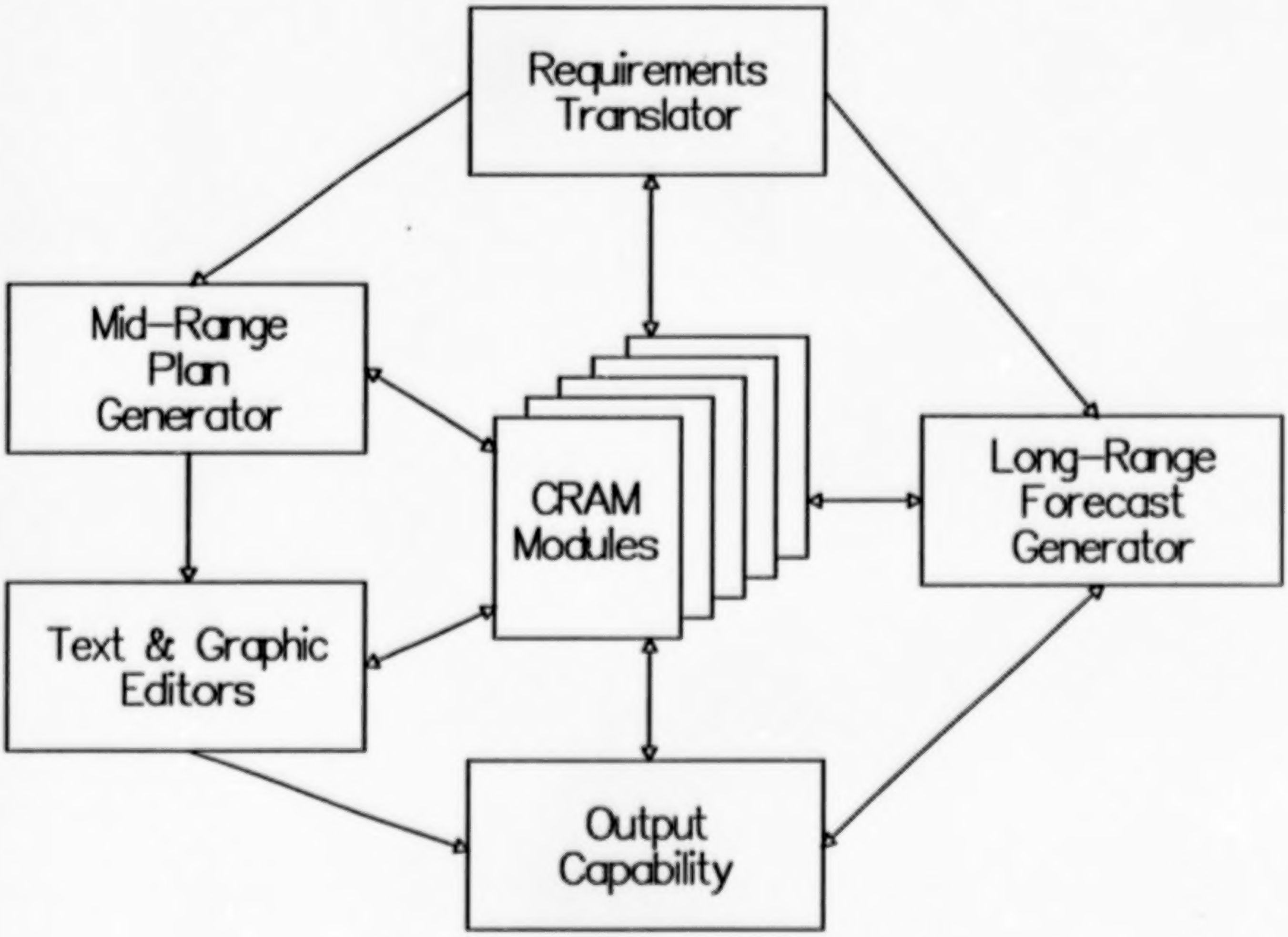


Figure 5 – RALPH System Architecture

Table 1 — Definitions of Resource Allocation Event Priorities

Priority	Activity Period and Priority Criteria*	Examples**
1	Spacecraft Emergency which Threatens Achievement of Primary Objectives Time Critical	Determined in Real Time
2	Single Opportunity or One-time Event Manditory for Achievement of Primary Objectives Time Critical	Launch Midcourse Maneuvers Planetary Near Encounter Some Scientific Events**
3	Irregular Events with Subsequent Opportunity Available Mandatory for Achievement of Primary Objectives Time Critical	Trim Maneuvers Some Orbital Cruise** Some Interplanetary Cruise** Planetary Radar
4	Regular or Repeated Events Mandatory for Achivement of Primary Objectives Time Critical	Telemetry Dumps Some Interplanetary Cruise** Some Orbital Cruise**
5	Not Mandatory for Achievement of Primary Objectives Time Critical	Extended Mission Some Interplanetary Cruise** Planetary Radio Astronomy Some Orbital Cruise**
6	Mandatory for Achivement of Primary Objectives Not Time Critical	Some Orbital Cruise** Some Interplanetary Cruise** Pulsar Rotation Constancy
7	Not Mandatory for Achivement of Primary Objectives Not Time Critical	"Priority-6" for Extended Mission

*These criteria are subject to revision by the RAPT, but they have not been revised for a number of years.

**Actual events as governed by the priority criteria would, of course, be more project-specific.

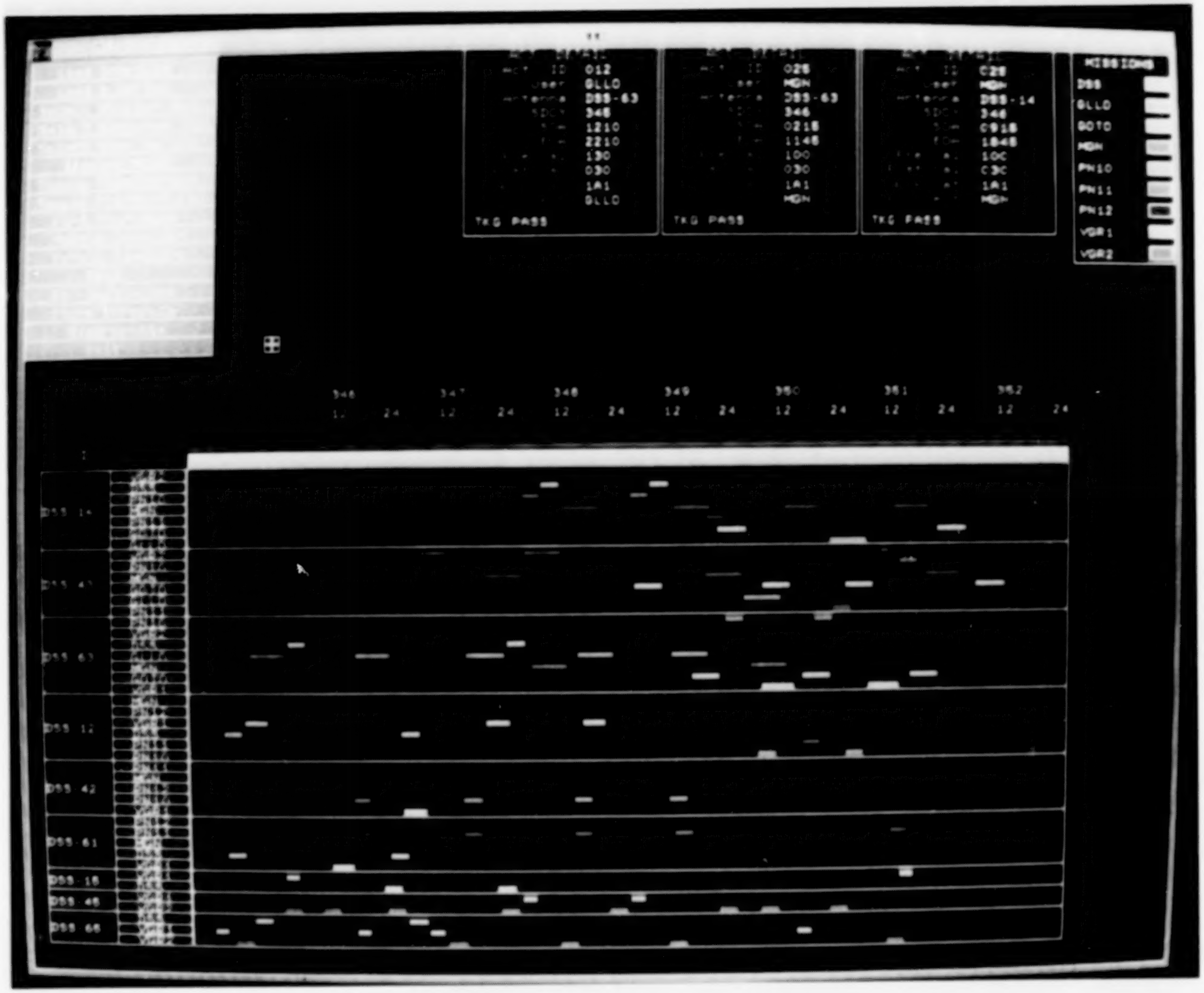

MISSIONS
DSS
GLLO
GOTO
MGN
PN10
PN11
PN12
VGR1
VGR2
TKG PASS
346
347
348
349
350
351
352
12
24
DSS 14
DSS 43
DSS 63
DSS 12
DSS 42
DSS 61
DSS 15
DSS 45
DSS 65

Figure 6 – The RALPH Graphic Editor Allows Interactive Fine-tuning of Plans

BEST COPY AVAILABLE

Table 2 – Results from RALPH Implementation

Item	1986	1989
Mid-range plan	25 hours	5 hours (Savings: 1040 hrs/yr)
Meetings per week	3-4	1 (Savings: 6240 hrs/yr)
Planning horizon	3 months	10 years
Long-range forecast	Impossible	Possible
User interface	Informal	Standardized
Requirements	Specific	Generic
Data dissemination	Manual	Electronic
Quick-look studies	Very difficult	Now doing one/week

The RAPT meetings have benefitted as a result of more efficient output products. Previously, three to four negotiation meetings were required each week to clear conflicts. The research process of formalizing the event model led to a well-defined interface structure between resource users and operations personnel. Even without any software implementation, this increased definition has led to a significant reduction in requirements generation and negotiation time among users. The addition of the RALPH tool has further reduced the conflict level to the extent that one meeting per week is sufficient (Table 2).

There are several key factors that contributed to the successful implementation of the automated software tool. The formation of a Design Team, which merged operational users and software developers, was a catalyst for a better global understanding of the functionality that is the core of the resource allocation process. The merging of knowledge engineering and operations research methodologies, with the application of the five-rule paradigm, clearly defined the operational environment. All of these factors resulted in the creation of **effective** software.

The RALPH software is highly data driven and modular. These features, combined with the generic designs of the data structures, make the RALPH system extremely easy to modify. Object attribute representations are readily updated by operations personnel to reflect a changing resource environment without modification of the programs that use the data. The data structure and software modularity enable RALPH to cross problem boundaries and support related tasks with minimal development effort. These features enable the Resource Analysis Team to respond quickly to requests for special studies and management reports. A typical statistical report output from the RALPH system is depicted in Figure 7.

The evolution provided by the RALPH system from a forecast to an operational plan has provided an efficient utilization of limited ground data system resources. RALPH has affirmed the significance of establishing and applying event priorities based on scientific merit and the benefit of continuity in planning provided by knowledge-based engineering. RALPH is designed as an expandable system to meet the needs of an evolving allocation process. With appropriate emphasis on problem understanding and representation, the same methodologies merged to build RALPH can be utilized to support a large class of resource allocation and planning systems. The RALPH system exhibits strong potential for minimizing development cycles of resource and payload planning systems throughout NASA and the private sector.

Reference

Craig D. Johnson and David G. Werntz, "A New Methodology for Resource Allocation and Planning", International Space Conference, Johnson Space Center, Houston TX, November 17, 1987.

Acknowledgements

The work described in this paper was performed at the Jet Propulsion Laboratory, California Institute of Technology, and was funded by the JPL Flight Project Support Office under contract with the National Aeronautics and Space Administration.

About the Authors

Carol A. Berner is the Cognizant Operations Engineer of the Resource Analysis Team. Her nine years of experience at JPL cover many aspects of telecommunications and space flight operations. A member of IEEE, she has a B.S.E.E. from the University of California at Los Angeles, and is currently working on an M.S.E.E. specializing in the field of Intelligent Systems at the University of Southern California.

Ralph Durham is the Resource Analysis Team Leader. He contributes 13 years of experience at JPL in Resource Planning and Analysis.

Norman B. Reilly is the RALPH System Engineer. His B.S.E.E. is from Columbia University, his M.S.E.E. from the Polytechnic Institute of New York. He has over 20 years of experience in computer systems and systems engineering.

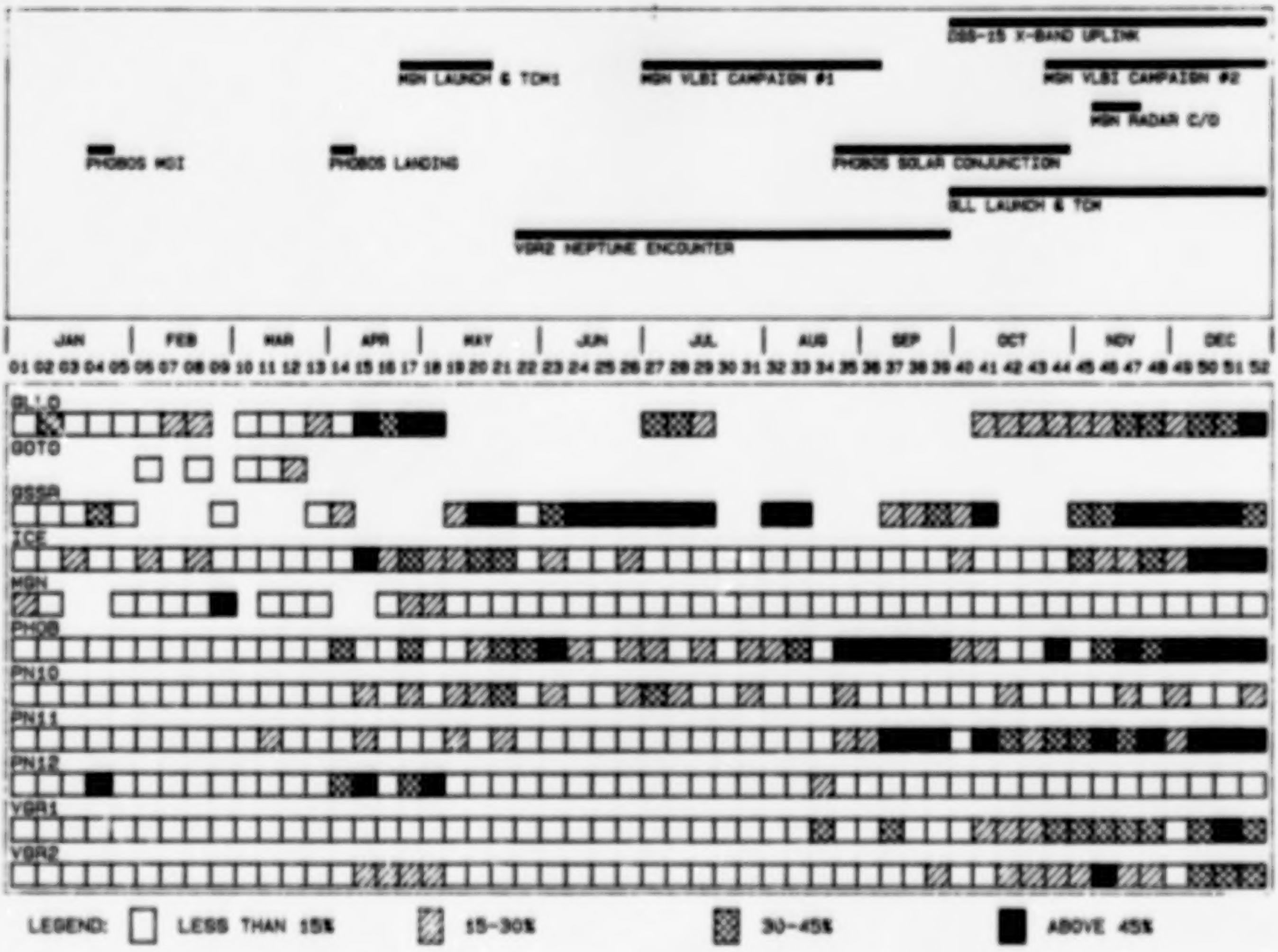


Figure 7 – Example of output: contention by user

BLANK PAGE

A Situated Reasoning Architecture for Space-based Repair and Replace Tasks[*]

Ben Bloom
Debra McGrath
Jim Sanborn

The MITRE Corporation
7525 Colshire Drive
McLean, VA 22102-3481

Abstract

An area of increasing interest within AI and Robotics is the integration of techniques from both fields to the problem of controlling autonomous systems. Space-based systems, such as NASA's EVA Retriever, provide complex, realistic domains for this integration research. Space is a dynamic environment, where information is imperfect, and unexpected events are commonplace. As such, space-based robots need low level control for collision detection and avoidance, short-term load management, fine-grained motion, and other physical tasks. In addition, higher level control is required to focus strategic decision making as missions are assigned and carried out. Throughout the system, reasoning and control must be responsive to ongoing change taking place in the environment.

This paper reports on current MITRE research aimed at bridging the gap between high level AI planning techniques and task-level robot programming for telerobotic systems. Our approach is based on incorporating *situated reasoning* into AI and Robotics systems in order to coordinate a robot's activity within its environment. Thus, the focus of this research is on controlling a robot embedded in an environment, as opposed to the generation and execution of lengthy robot plans. We present an integrated system under development in a "component maintenance" domain geared towards repair and replacement of Orbital Replacement Units (ORUs) designed for use aboard NASA's Space Station *Freedom*. The domain consists of a component-cell containing ORU components and a robot (manipulator and vision system) replacing worn and/or failed components based on the collection of components available at a given time. High level control reasons in "component space" in order to maximize the number operational component-cells over time, while the task-level controls sensors and effectors, detects collisions, and carries out pick and place tasks in "physical space." Situated reasoning is used throughout the system to cope with, for example, non-deterministic component failures, the uncertain effects of task-level actions, and the actions of external agents operating in the domain.

---

[*] This work is funded by MITRE-Sponsored Research Projects 97060 and 97140. The authors wish to thank MITRE for its ongoing support of this work. Email address: sanborn@ai.mitre.org.

1 Introduction

There is a resurgence of interest within the AI and Robotics communities in integrated efforts leading to the development of robust, autonomous systems for use in dynamic, uncertain, and unpredictable domains. NASA in particular has several efforts underway, including its *Systems Autonomy Technology Program* (SATP), a ten year program to establish NASA as a world leader in intelligent autonomous systems research and development, the EVA Retriever, and the Mars Rover project. These programs are aimed at addressing two issues in space exploration:

1. For manned missions, human EVA is dangerous, expensive, and time-consuming.
2. For unmanned missions, signal delay times require autonomous control throughout non-trivial time intervals.

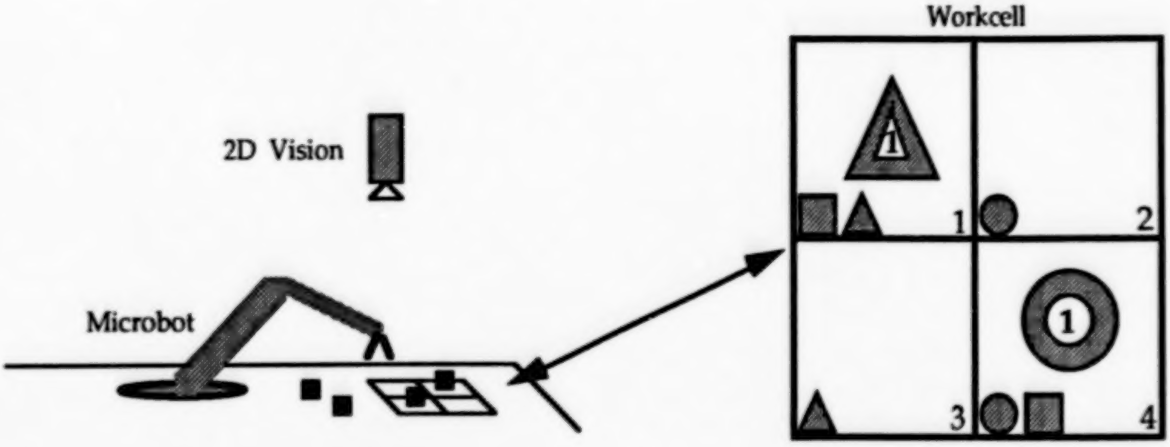
One of the major hurdles in building autonomous systems is the integration of off-line deliberative reasoning (e.g., task planning, route planning, and resource allocation, etc.) with real-time situated control (e.g., collision and obstacle avoidance, path expansion, load management, calibration from landmarks, etc.). The former type of reasoning is *goal-directed*, and has led to the development of several constraint-posting planners (e.g., [Ste81], [Wil84], [Cha85]) generating plans to satisfy multiple goals. The latter type of reasoning is *event-driven*, in that responses to existing, perceived, or projected situations are required in order to maintain overall system integrity. AI research has begun to address situated reasoning, as well its integration with off-line plan generation (see, for example, [Kae86], [GLS87], [AC87], [Dea87], and [SH88]).

This paper presents the initial results of a combined MITRE research effort integrating AI planning and situated reasoning techniques with task-level robotics and perception for space-based autonomous systems. The long-term goal of this research is to integrate off-line planning, situated reasoning, and sensor/actuator subsystems across various levels of abstraction in order to provide both the reactive behavior necessary for survival in realistic environments, and the introspective reasoning required to carry out deliberate tasks and achieve desired goals. The work presented in this report lays the groundwork for this long-term goal by providing an integrated situated reasoning and task-level control architecture, as well as a system operating in a realistic application domain. Examples from this domain are used throughout the paper to illustrate the approach.

2 A Component Repair and Maintenance Domain

One of the many application areas for space-based autonomous systems is routine extravehicular maintenance. By "routine maintenance," we refer to a general class of situations in which components of a system are scheduled for maintenance (as determined by expected lifetime) and are also tended to when they fail unexpectedly. In such an application, a robot must allocate available resources (spare parts, or modular components such as ORUs) in order to maximize the overall operating status of a collection of components.

The "routine repair and replace" domain shown in Figure 1 captures this idea. It consists of a robotic manipulator, vision system, a *component workcell*, and a collection of *components*. The workcell is an $N \times M$ array of *compartments*,



(a) Physical Space

(b) Component Space

Figure 1: A Simple Component Repair and Maintenance Domain

each of which may be "filled" by components of various types. In this way, the workcell may be thought of as a modular "breadboard" into which components are inserted to become operational. Each workcell compartment is labelled according to the types of components that may fill it. A component is said to be *acceptable* to a compartment whenever it may be used to fill that compartment. Finally, an expected lifetime (the *mean time to failure*, MTTF) is associated with each type of component in order to model both routine and unexpected component failures.

In the current system, the workcell is modelled by a "bin" on a tabletop, with different shaped objects (cylinders, rectilinear and triangular blocks, etc.) representing various component types (see Figure 1(a)). States of the domain are subject to constant flux at the hands of external (human) agents, whose unanticipated actions may include adding, removing, moving, and breaking components. In addition, the fact that components may fail unexpectedly at any time also requires attention to the ongoing situation. The choice of this repair and replace domain was influenced by the following considerations:

1. The architecture should be realistically scalable to handle any of a variety of repair and replace tasks to be performed in environments characterized by dynamics and uncertainty (such as Space Station ORU replacement).
2. The scenario requires the integration of physical control (robotics) with high-level reasoning (AI).
3. The hardware required for developing the testbed scenario was readily available.

2.1 Physical vs. Component Space

The reasoning required for successful operation in this domain falls into two classes: reasoning in *physical space*, and reasoning in *component space*. Physical space reasoning includes the planning, executing, and monitoring of collision-free paths for the manipulator and moved objects, detecting obstacles, noticing objects when they are moved, and generally dealing with physical aspects of the domain. The physical space reasoner used to control the manipulator and vision system shown in Figure 1(a) is known as the *Task-level Robot Programming System* (TLRPS). Component space reasoning includes allocating available components among empty compartments, prioritizing replacement and repair tasks according to various heuristics, as well as reacting to unexpected component failures, moves, additions, and deletions. The *Component Space Reasoner* (CSR) provides this functionality for the component space corresponding roughly to Figure 1(b).

2.2 Interface Language

This section presents the communication specification between TLRPS and CSR. The interface has been designed to distinguish between physical and component space aspects of the domain.

2.2.1 CSR→TLRPS

`Put_in` *object compartment*$_i$: Move *object* from its present (table) location into the (empty) *compartment*$_i$.

`Put_at` *object* x y : Place *object* on the tabletop at TLRPS coordinates (x y); *object* is assumed to be either on the table or within the workcell.

`Put_down` *object* x y : Put down (held) *object* on the tabletop at TLRPS coordinates (x y).

2.2.2 TLRPS→CSR

`Begin_update` : Initiates an update of object and location information from TLRPS. Followed by one or more instance of:

`Delete` *object* : *object* has been removed from the domain.

`Move` *object* x y θ : *object* now centered at (x y) rotated by θ degrees.

`Add` *object* x y θ : *object* has appeared in the domain, centered at (x y) rotated by θ.

`End_update` : Signals the end of the update.

`Failed_grasp` *object* : TLRPS could not grasp *object*.

`Collision` *held_object object_in_path* : *object_in_path* prevents moving *held_object*.

`Unreachable_object` *object* : *object* cannot be reached in its current location.

`Unreachable_location` x y : (x y) cannot be reached.

3 Technical Approach

This section describes the operation of TLRPS, CSR, and their integration in the repair and replace domain. Since this domain is non-static, each system must cope with discrepencies between anticipated and actual states of the domain. For example, components may move from expected locations and may fail (or be broken) before their MTTF has elapsed. Since external agents may change the environment, neither system can make accurate long-term projections regarding future states. Rather, the system must optimize local behavior based on existing and projected states given the overall component maintenance goals. The top-level system architecture is shown in Figure 2.

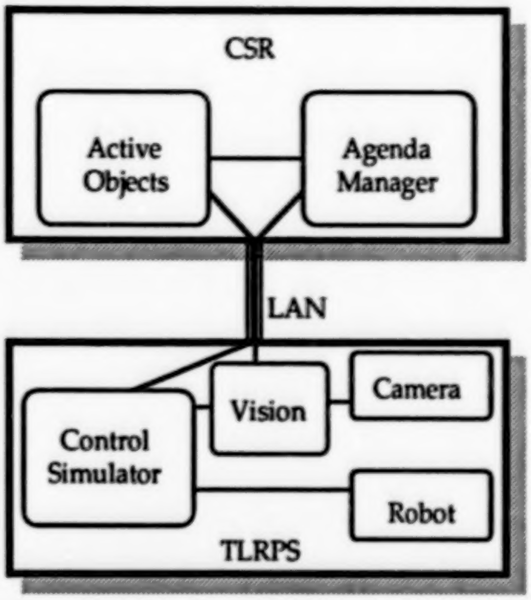


Figure 2: Top-level System Architecture

3.1 Physical-Space Reasoning

TLRPS resides on a Silicon Graphics IRIS 4D/70GT workstation running the IGRIP 3D robotics modeling and simulation system from Deneb Robotics. The vision system consists of software provided by NASA running on an IBM PC-AT with added frame grabber and image processing boards from Data Translation. The robot currently in use is a Microbot Alpha I.

A single camera with a fixed viewpoint is used to capture the layout of the robot's workspace. The vision software in the PC-AT classifies the objects in the workspace according to its training data set corresponding to the physical component objects. A workspace description containing object types and their locations and orientations is then sent to the IRIS. Image processing software on the IRIS interprets the workspace description on each cycle and modifies the world model accordingly, generating CSR update messages, as well as updating the IGRIP simulation's 3D graphic display of the robot and the workspace. Users may interact directly with TLRPS by entering task-level commands via pop-up menus, or turn control over to CSR. When a task-level command is received by TLRPS (whether from CSR or a human user), the command is simulated in 3D graphics and executed by the robot. The simulation runs one step ahead of the actual robot execution, checking for possible collisions and out-of-reach conditions. If such a condition is detected, TLRPS performs error recovery operations, registering the simulation and the robot to a safe configuration. An exception-dependent error message is generated and sent to CSR when it is controlling TLRPS.

3.2 Component-space Reasoning

CSR is divided into two main modules: an *agenda manager* for prioritizing tasks and issuing commands to TLRPS, and a collection of *objects* corresponding to the various physical objects in the domain at a given time. CSR's top-level control loop is

```
(0) get component types used
(1) await TLRPS update
(2) process TLRPS update
(3) if there is an executable task
(4)     then issue it to TLRPS
(5) goto (1)
```

The initialization step (0) determines components type definitions and their associated parameters. This is accomplished by consulting a file provided before startup. Once this information has been processed, CSR enters its main control loop. The first stage of this loop (Step 1 above) simply puts CSR into a wait state until an update from TLRPS is available. Recall that these updates consist of a collection of **Add**, `Delete`, and `Move` messages. The second stage (Step 2) processes the update's content. During this stage, messages are sent to existing CSR objects mentioned in the update; the message sent depends on what the update to the object happens to be. During the processing of these messages, objects may request that the agenda manager generate new tasks to, or remove existing tasks from, its collection of tasks. In the final stage (Steps 3 and 4), the agenda manager prioritizes its collection of tasks and, if any are executable, issues a command to TLRPS to carry out the most important executable task. Within CSR, a *task* is one of:

`routine_replace` *component* *compartment*$_i$: replace worn (but operating) *component* with an existing new component acceptable to *compartment*$_i$.

`immediate_repair` *component* *compartment*$_i$: replace failed *component*.

`await_component` *compartment*$_i$: fill *compartment*$_i$ when an acceptable component is added.

`install_component` *component* *compartment*$_i$: put *component* into *compartment*$_i$ and start it operating.

`deinstall_component` *component* *compartment*$_i$: stop operating *component* and remove it from *compartment*$_i$.

`move_component` *component* x y : move *component* from its current location to $(x\ y)$.

3.2.1 Active Objects

Most of the reasoning done by CSR is triggered during update processing. For each physical object in the domain, CSR generates a *component* object in its component space. The workcell, and each of its compartments, is also represented as an active CSR object. Updates from TLRPS relating to physical objects are then translated into messages to CSR objects, which may take various update-dependent actions. This approach associates reasoning with changing information at the object level, rather than on the more traditional rule-interpreter/database approach. As such, reasoning and control are modified by changing objects' responses to messages, rather than by the actions contained in rule consequents.

To illustrate this approach to object modelling and situated reasoning, we follow the processing of CSR's initial update. This update consists of a collection of `Add` messages from TL-RPS. The first such message processed corresponds to the workcell. On encountering this message, CSR generates a workcell object, an object for each of the workcell's compartments, and initializes their parameters. Following this, a component object is created for each added component. As part of its `Add` message processing, the component checks to see whether or not it lies within some compartment. If not, its status is "new." Otherwise, the compartment object is consulted to determine whether or not the component should be accepted. If so, the compartment "installs" the component, by setting the component's status to "operating," and schedules a `routine_replace` task for sometime in the future, depending on the MTTF of the newly installed component. If the compartment does not accept the component, it attempts to have the offending component removed by generating a `move_component` task.

Once this initial update has been processed, CSR knows which compartments contain operating components, which need to be filled, as well as which components are available in the domain. Empty compartments generate `await_component` tasks and compete for newly arriving components according to several criteria, including (1) how long the compartment has been empty, and (2) the number of different types of components the compartment accepts. The next section discusses the use of these heuristics in prioritizing tasks.

3.2.2 Tasks and the Agenda Manager

In order to take action, component and compartment objects generate tasks which are added to CSR's *agenda*: a simple partially ordered set of tasks. All tasks have the following properties, which are used in determining their relative importance and/or execution status:

`status` : one of

- `:pending` : not yet ready for execution,
- `:executable` : ready for execution,
- `:active` : in progress, or
- `:done` : finished, does not indicate success or failure.

`priority` : a measure of the task's importance, one of

- `:normal` : a routine task, such as `routine_replace`,
- `:asap` : a non-routine task, such as `immediate_repair`, or
- `:now` : highest priority tasks, such as `await_component`.

`resource_measure` : measure of difficulty in obtaining resources for this task.

`timestamp` : actual task instantiation time.

`supertask` : associated parent task.

`subtasks` : subtasks comprising an abstract task.

Tasks fall into two classes: *primitive* tasks, corresponding directly to TLRPS commands, and *abstract* tasks, which have no analogous TLRPS command. Abstract tasks may have an associated *test* executed once per cycle whenever the task's status is `:pending`. All tasks have an *act*, which is executed when an `:executable` task becomes `:active`. In the current system, `install_component`, `deinstall_component`, and `move_component` are primitive (corresponding to various types of `put_in`, `put_at`, and `put_at` commands, respectively); all other CSR tasks are abstract.

In most traditional plan generation and execution systems, a complete plan to achieve a goal is generated and then executed stepwise. An underlying assumption of this approach is that the world will behave as expected during plan execution. If exceptions occur during execution, the usual recourse is to more planning. As an example, a "routine replace" operation on a compartment is normally composed of two steps:

1. `deinstall_component` *oldc* $\mathit{compartment}_i$
2. `install_component` *newc* $\mathit{compartment}_i$

under the assumption that *newc* will be available when step (2) is to be executed. However, if *newc* is removed sometime during the execution of step (1), the resulting situation is the same as one in which $\mathit{compartment}_i$ were simply waiting to be filled. Recall that in this case, the empty compartment generates an `await_component` task to find and then install a suitable component.

In general, CSR uses component and compartment objects to assist in carrying out plans whenever the domain is cooperative, but also ensures that the appropriate behavior results when "assumptions" fail. Rather than planning steps for anticipated future states, CSR generates one step at a time, and uses feedback from the world to determine its next step. The definition for `routine_replace` looks like

```
(deftask routine_replace
         (compartment
          component
          new_component
          replace_time)
  :test (and (> (now) replace_time)
             new_component)
  :act (generate_task
          'deinstall_component
          component compartment)
       (reserve compartment
                new_component))
```

Notice that there is no mention of a task corresponding to step (2) above in this definition. If all goes well in the world, the compartment will generate an `install_component` using the new component, reserved for it by the `routine_replace` task, once the deinstall has been successfully carried out. However, if for some reason the new component is no longer available, the compartment simply generates an `await_component`, which searches for another suitable replacement. Since no assumption is made as to whether or not the new component will remain available, the appropriate response occurs in either case. This approach requires that CSR objects track their allocations (so that, for instance, the compartment can determine whether or not its reserved component is available or not), but this is easily managed by informing objects of their allocations and taking appropriate action during update message processing.

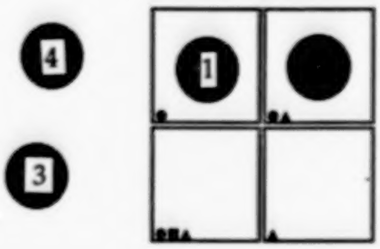
The task agenda is ordered in decreasing order of importance. This ordering is maintained by merging new tasks, and re-merging tasks when their parameters change, according to the following sequence of pairwise tests:

1. status – `:executable > :pending`
2. priority – `:now > :asap > :normal`
3. resources – prefer more constrained
4. time – prefer older tasks
5. arbitrary

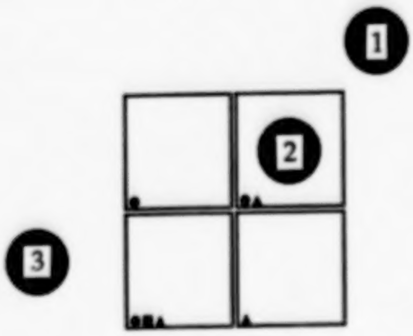
One case where re-merging is necessary is when an acceptable component becomes available to a pending `await_component` task. New components announce their availability by sending an `advertise-available-resource` message to the agenda manager, which in turn offers the new resource to pending tasks in decreasing order of importance. If a task allocates the available resource, the resource is allocated to it, and it is remerged into the agenda. When an available component is offered to an `await_component` task, it is allocated so long as it is acceptable to the task's compartment.

Another way an `await_component` task may find an acceptable component is for it to ***usurp*** the resources of another, less important task. In general, this process examines the task agenda from back to front until either (1) an acceptable resource is found, in which case it is usurped and the two tasks are re-merged into the agenda, or (2) the process reaches a task with higher priority than the intented usurper's, in which case no suitable resource is available.

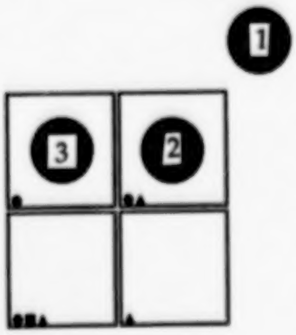
Figure 3(a) shows the state of a workcell as `cylinder_1`, in *compartment*$_1$, is to



(a) deinstall



(b) usurp



(c) install

Figure 3: CSR in action.

be replaced with `cylinder_4`. The work-cell also contains a cylinder (`cylinder_2`) operating in *compartment*$_2$ and scheduled for replacement by `cylinder_3` at some later time. The `routine_replace` task associated with *compartment*$_1$ acts, generating a `deinstall_component` task, which is placed at the head of the agenda (since it is primitive, and hence, immediately executable). This in turn issues a TLRPS put_at command to remove `cylinder_1`. Sometime during the execution of this command, `cylinder_4` is removed, resulting in the situation shown in Figure 3(b). At this point, since *compartment*$_1$ is empty, and no longer has a reserved component, it generates an `await_component` task. `Cylinder_3` should be "available" to this new task, but it is currently allocated to the `routine_replace` task associated with *compartment*$_2$. However, since the new task has higher priority, it usurps the resources (`cylinder_3`) of the lower priority `routine_replace` task. This changes its status to `:executable`, its action generates a primitive `install_component` task, which in turn issues a TLRPS put_in command to complete the replacement, as shown in Figure 3(c).

3.3 CSR/TLRPS Integration

The two systems presented above have been developed jointly but at physically different sites in McLean, Virgina (CSR), and Houston, Texas (TLRPS). CSR is implemented in Portable Common Loops (PCL) and resides on a Symbolics Lisp Machine. As noted above, TLRPS resides on a Silicon Graphics machine. The two systems are connected on an Ethernet LAN at the Houston site. They communicate via TCP/IP streams over this network, using the interface language presented in Section 2.2.

3.4 The TLRPS Simulator

Since CSR and TLRPS have been developed at different sites, a TLRPS simulator has been implemented for CSR's development and testing. In addition to simulating TLRPS's physical-space reasoning, allowing external agents to manipulate the domain, this simulator models component operation so that unexpected (i.e., pre-MTTF) component failures may occur. It also provides a graphic user-interface to CSR.

The TLRPS simulator used in testing CSR also runs on a Symbolics Lisp Machine, with the two systems simulating CSR↔TLRPS interface over a local Chaosnet. Figure 3 was generated using screen images from the TLRPS simulator.

4 Current Status and Future Work

This integrated project has addressed the problem of integrating high-level and task-level reasoning in a dynamic environment. The architectures used in both systems are domain-independent, and will be useful for other NASA applications, as well as broader application in manufacturing and assembly, hazardous materials handling, military operations, and undersea work.

The existing system is able to react to any unexpected change that occurs during the execution of a single CSR primitive task. During periods of CSR inactivity, updates are available at a rate of approximately one every ten seconds. CSR's real response time is on the order of one-tenth of a second, so the "snapshot" nature of updates and update processing suffices for this domain. In general, the rate of change in dynamic domains is much faster, so that situated reasoning must be based on *projecting* future states in order to anticipate and avoid exception situations. An approach to situated reasoning based on these observations is presented in [San88].

Due to existing hardware, the current system has very little low-level reactive capability. An improved hardware system and more integrated reasoning and control architecture will be required for more general purpose, robust autonomous control. To this end, MITRE is establishing an *Autonomous Systems Laboratory* (ASL). Research in the ASL will focus on the integration of deliberative (off-line) planning, situated reasoning, and hardware subsystems. The ASL will be composed of "off the shelf" sensing, robotics, and AI hardware and firmware representing the significant advances made in these technologies in recent years. The focus of the current project will shift from situated reasoning under a constant goal (e.g., routine repair and replace), toward more flexible control in a domain where several different types of goals are to be achieved over time. A ground-based mobile system operating in a dynamic domain will be used as a test-bed to simulate a flexible space-based automaton for routine extra-vehicular repair, assembly, and retrieval tasks. Deliberative planning will take as input a collection of tasks to be carried out (the "daily schedule") and determine an ordering among these tasks. Its output will be information used to monitor activity and constrain low-level task execution in the domain via a situated reasoning system. This latter system actually controls the physical system as it operates in its environment by controlling its reactions to existing and anticipated states of affairs. This on-line system uses the constraints from the deliberative planner as heuristics in selecting among tasks it can perform, but is independently capable of a basic level of competence in the domain.

Acknowlegements

The authors thank Pete Bonasso and Jim Reynolds for their contributions to this work, as well as useful comments on this paper. In addition, productive discussions have been held with Jack Benoit, Chris Elsaesser, and Vincent Hwang.

References

[AC87] Philip Agre and David Chapman. Pengi: an implementation of a theory of action. In *Proc. AAAI-87*, 1987.

[Cha85] David Chapman. *Planning for Conjunctive Goals.* AI 802, MIT, 1985.

[Dea87] Thomas Dean. Planning, execution, and control. In *Knowledge-Based Planning Workshop*, pages 29–1–29–10, DARPA, 1987.

[GLS87] Michael Georgeff, Amy Lansky, and Marcel Schoppers. *Reasoning and Planning in Dynamic Domains: An Experiment with a Mobile Robot.* International Technical Note 380, SRI, 1987.

[Kae86] Leslie Pack Kaelbling. An architecture for intelligent reactive systems. In M. Georgeff and A. Lansky, editors, *The 1986 Workshop on Reasoning about Actions and Plans*, pages 395–410, Morgan Kaufman, 1986.

[San88] James Sanborn. *A Model of Reaction for Planning in Dynamic Environments.* Master's thesis, University of Maryland, May 1988.

[SH88] James Sanborn and James Hendler. A model of reaction for planning in dynamic environments. *International Journal of AI in Engineering*, 1988. Special issue on planning.

[Ste81] M.J. Stefik. Planning with constraints (molgen: part 1). *Artificial Intelligence*, 16:141–169, 1981.

[Wil84] D.E. Wilkins. Domain-independent planning: representation and plan generation. *Artificial Intelligence*, 22:269ff, 1984.

BLANK PAGE

Parallel Plan Execution with Self-Processing Networks

C. Lynne D'Autrechy and James A. Reggia*

Department of Computer Science
University of Maryland, College Park, MD 20742

*Also with the University of Maryland Institute for Advanced Computer Studies
and the Department of Neurology, University of Maryland

Abstract: A critical issue for space operations is how to develop and apply advanced automation techniques to reduce the cost and complexity of working in space. In this context, it is important to examine how recent advances in self-processing networks can be applied for planning and scheduling tasks. For this reason, we are currently exploring the feasibility of applying self-processing network models to a variety of planning and control problems relevant to spacecraft activities. Our goals are both to demonstrate that self-processing methods are applicable to these problems, and that MIRRORS/II, a general purpose software environment for implementing self-processing models, is sufficiently robust to support development of a wide range of application prototypes. Using MIRRORS/II and marker passing modelling techniques, we implemented a model of the execution of a "Spaceworld" plan which is a simplified model of the Voyager spacecraft which photographed Jupiter, Saturn, and their satellites. This study demonstrates that plan execution, a task usually solved using traditional AI techniques, can be accomplished using a self-processing network. The fact that self-processing networks have been applied to other space-related tasks in addition to the one discussed here demonstrates the general aplicability of this approach to planning and control problems relevant to spacecraft activities. This work also demonstrates that MIRRORS/II is a powerful environment for the development/evaluation of self-processing systems.

I. Introduction

A critical issue for space operations is how to develop and apply advanced automation techniques to reduce the cost and complexity of working in space. In this context, it is important to examine how recent advances in self-processing networks (connectionist models, artificial neural networks, marker passing systems, etc. [3]) can be applied to planning and scheduling tasks. Most successful work with such models has focused on fairly low-level applications (pattern recognition or completion, associative memory, constraint satisfaction, etc.) and relatively little has been done in traditional AI problem-solving areas like planning. Thus, while these methods potentially offer tremendous advantages for complex automation applications (massively parallel processing, fault tolerance, etc.), it is currently difficult to see how they can be adopted directly.

For this reason, we are exploring the feasibility of applying self-processing network models to a variety of planning and control problems relevant to spacecraft activities. Our goals are both to demonstrate that self-processing methods are applicable to these problems, and that MIRRORS/II, a general purpose software environment for implementing self-processing models [1,2], is sufficiently robust to support development of a wide range of application prototypes. While a number of specific applications have recently been developed using MIRRORS/II for spacecraft applications (camera controller [5], diagnostic problem-solver [6], etc.), this paper focuses on a specific plan execution example.

II. Self-Processing Network Models and Marker Passing

To enable the reader less familiar with current work on self-processing network models to follow the principal ideas embodied in MIRRORS/II, we introduce some basic concepts and terminology, simplifying somewhat for brevity. The term *self-processing network model* as used in this paper refers to models in which many, usually simple, processing elements operate in parallel and communicate via connections (links) between nodes. For our purposes, it is convenient to view self-processing network models as having two components: a network and an activation method. The *network* consists of a set of processing *nodes* connected together via *links*. Nodes directly connected to one another are said to be *neighbors* of each other. The *activation method* is a *local* rule or procedure that each node follows in updating its current state in the context of information from neighboring nodes. Typically, the goal in constructing and running a simulation with a self-processing network model is to demonstrate that some *global* behavior (behavior of the network as a whole) can emerge from the concurrent *local* interactions between neighboring nodes during a simulation.

A great number of self-processing network models have been proposed and studied since the 1940's in cognitive science, artificial intelligence, and neurophysiological modelling [3]. This paper is only concerned with one class of such models referred to as *marker passing systems*. The networks in these symbol-processing models usually have semantically-labeled, unweighted links and implement spreading activation by passing symbolic (non-numeric) labels or markers. Typically, a node might have a dozen marker bits (*M*1, *M*2, ..., *M*12), binary switches that can be turned on or off when the appropriate marker is "received." Using these markers, such networks provide powerful mechanisms for implementing set operations (intersection, union, etc.) as well as some forms of deduction (transitive closure of relations, inheritance of properties, etc.).

III. The MIRRORS/II Simulator

MIRRORS/II is an extensible general-purpose simulator which can be used to implement a broad spectrum of self-processing network models. MIRRORS/II is distinguished by its support of a high-level non-procedural language, an indexed library of networks, spreading activation methods, learning methods, event parsers and handlers, and a generalized event-handling mechanism.

The MIRRORS/II language allows relatively inexperienced computer users to express the structure of a network that they would like to study and the parameters which will control their particular self-processing network model simulation. Users can select an existing spreading activation/learning method and other system components from the library to complete their model; no programming is required. On the other hand, more advanced users with programming skills who are interested in research involving new methods for spreading activation or learning can still derive major benefits from using MIRRORS/II. The advanced user need only write functions for the desired procedural components (e.g., spreading activation method, control strategy, etc.). A specification file, written in the MIRRORS/II language by the user, serves as input to MIRRORS/II.

Self-processing network models developed using MIRRORS/II are not limited to a particular processing paradigm. Many spreading activation methods and learning methods including Hebbian learning, competitive learning, and error back-propogation are among the resources found in the MIRRORS/II library. MIRRORS/II provides both synchronous and asynchronous control strategies that determine which nodes should have their activation values updated during an iteration. Users can also provide their own control strategies and have control over a simulation through the generalized event handling mechanism.

Simulations produced by MIRRORS/II have an event-handling mechanism which provides a general framework for scheduling certain actions to occur during a simulation. MIRRORS/II supports system-defined events (constant/cyclic input, constant/cyclic output, clamp, learn, display and show) and user-defined events. An event command (e.g., the input-command) indicates which event is to occur, when it is to occur, and which part of the network it is to affect. At run time, the appropriate event handler performs the desired action for the currently-occurring event.

MIRRORS/II was originally designed for implementing connectionist models with no significant thought being given to how marker-passing methods might be developed in the context of MIRRORS/II. Thus, at the start of the work described here, it was not immediately obvious whether MIRRORS/II could support marker passing methods without alterations.

IV. Non-Hierarchical Plan Execution

Using MIRRORS/II and marker passing techniques, we implemented a model of the execution of a "Spaceworld" plan as described in [4]. Spaceworld is a simplified model of the Voyager spacecraft which photographed Jupiter, Saturn, and their satellites. The specific Spaceworld plan used here describes the sequential and parallel sequence of steps (goals) which must be taken by the Voyager spacecraft in order to photograph two satellites and transmit the photographs to earth. Each goal in the plan has the following

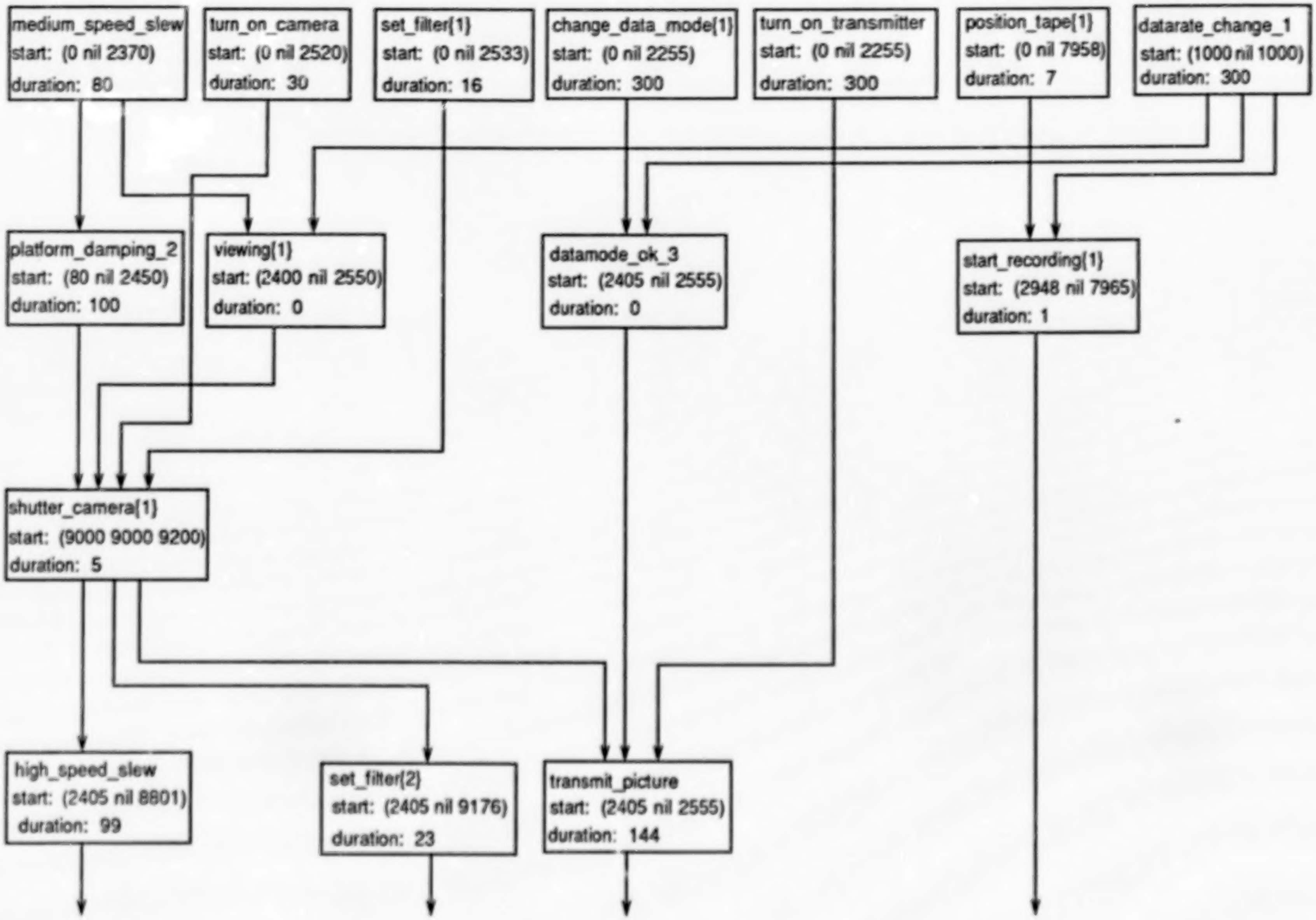

medium_speed_slew
start: (0 nil 2370)
duration: 80
turn_on_camera
start: (0 nil 2520)
duration: 30
set_filter{1}
start: (0 nil 2533)
duration: 16
change_data_mode{1}
start: (0 nil 2255)
duration: 300
turn_on_transmitter
start: (0 nil 2255)
duration: 300
position_tape{1}
start: (0 nil 7958)
duration: 7
datarate_change_1
start: (1000 nil 1000)
duration: 300
platform_damping_2
start: (80 nil 2450)
duration: 100
viewing{1}
start: (2400 nil 2550)
duration: 0
datamode_ok_3
start: (2405 nil 2555)
duration: 0
start_recording{1}
start: (2948 nil 7965)
duration: 1
shutter_camera{1}
start: (9000 9000 9200)
duration: 5
high_speed_slew
start: (2405 nil 8801)
duration: 99
set_filter{2}
start: (2405 nil 9176)
duration: 23
transmit_picture
start: (2405 nil 2555)
duration: 144

Figure 1: The beginning portion of Vere's non-hierarchical Spaceworld plan.

four parameters: an earliest starting time, an ideal starting time, a latest starting time, and a duration. Often, the ideal starting time parameter does not exist; in this case, the earliest starting time is also the ideal starting time. A portion of this specific Spaceworld plan is illustrated in Figure 1. Each box in the figure represents a goal in the plan. The first line in a box is the name of the goal. The second line labelled "start:" indicates the early, ideal, and latest starting times in seconds respectively. The third line labelled "duration:" indicates the duration in seconds. The arrows in the plan represent dependency relationships or the flow of execution. For example, the platform_damping_2 goal must finish executing before the shutter.camera{1} goal can begin executing.

A goal in the plan cannot be executed until two constraints have been satisfied - the dependency constraint and the starting time constraint. First, all goals which precede a given goal G in time must have finished executing before G can begin executing. Once this dependency constraint has been satisfied, the starting time constraint must be satisfied. Optimally, a goal should begin execution at its ideal starting time or, if that is not possible, at the earliest time thereafter up to and including the latest starting time. If the latest starting time is reached and the dependency constraint has not yet been satisfied then that goal will never be executed.

The MIRRORS/II specification of the self-processing network used to implement the execution of the Spaceworld plan discussed in [4] is pictured in Figure 2. Each goal in the plan is represented as a node in a self-processing network (all statements beginning with "[node" and ending with "]"). All the goals in the plan share the same parameters so their corresponding nodes are grouped into a single set (lines 1-22). Each node has two user-defined attributes - *start* and *duration*. The start attribute is a triple of values representing the earliest, ideal, and latest starting times for a goal in the plan. For example, the triple representing the earliest, ideal, and latest stopping times respectively for node shutter_camera{1} is (2400 2500 2550). All times in the plan are given in seconds relative to the starting time of the plan. The goal duration is a number representing the time in seconds it takes to complete the execution of the goal. Connections between the nodes represent the time sequence dependency relationships. For example, the statement [node turn_on_transmitter ... (plan transmit_picture)] in Figure 2 indicates that node turn_on_transmitter connects to node transmit_picture. This connection indicates that execution of the turn_on_transmitter goal must be completed before the execution of the transmit_picture goal can begin.

```
;--- Set containing all the goals (nodes) in a non-hierarchical Spaceworld plan

[set plan  (contains
        calibrate_gyros change_data_mode{1} change_data_mode{2}
        change_data_mode{3} change_data_mode{4} clear_tape
        consolidate_tape{1} consolidate_tape{2} datamode_ok_1
        datamode_ok_2 datamode_ok_3
        datamode_ok_4 datamode_ok_5 gyros_rev_up high_speed_slew
        medium_speed_slew platform_damping_1 platform_damping_2
        playback{1} playback{2} position_tape{1} position_tape{2}
        position_tape{3} record_picture roll set_filter{1} set_filter{2}
        shutter_camera{1} shutter_camera{2} start_recording{1}
        start_recording{2} transmit_picture
        turn_off_camera turn_off_gyros turn_off_heaters
        turn_off_tape_recorder turn_on_camera turn_on_gyros
        turn_on_heaters turn_on_transmitter viewing{1} viewing{2}
        begin_earth_occultation end_earth_occultation
        datarate_change_1 datarate_change_2 datarate_change_3)

        (initact 0.0)
```

Figure 2: An abridged MIRRORS/II specification file for the self-processing network of the complete Spaceworld plan. Some details were omitted for brevity. (Page 1 of 3.)

```
        (attribute start dynamic optional)      ;--- Definition of node attributes needed for this model.
        (attribute duration dynamic optional)
        (attribute type dynamic)
        (method plan)                           ;--- Marker passing activation method for
                                                ;--- non-hierarchical plans.
        (connects (plan oto incoming optional)] ;--- Nodes in this set connect to other nodes in this set.

;--- The following node statements give detailed node attribute information for
;--- each node as well as describing the node connections in the network.
;--- Duration times and start times are given in seconds.

[implicit (member plan)]

[node calibrate_gyros (type action)]
[node change_data_mode{1} (start (0 nil 2255.76))(duration 300)(type action)
        (plan datamode_ok_3)]
                .
                .
                .
[node datamode_ok_3 (start (2405.76 nil 2555.76))(duration 0)(type inference)
        (plan transmit_picture)]
[node datamode_ok_4 (type inference)]
[node datamode_ok_5 (type inference)]
[node gyros_rev_up (type event)]
[node high_speed_slew (start (2405.76 nil 8801))(duration 99)(type action)
        (plan (/ platform_damping_1 viewing{2}))]
[node medium_speed_slew (start (0 nil 2370))(duration 80)(type action)
        (plan (/ platform_damping_2 viewing{1}))]
[node platform_damping_1 (start (2504.76 nil 8000))(duration 300)(type event)
        (plan shutter_camera{2})]
[node platform_damping_2 (start (80 nil 2450))(duration 100)(type event)
        (plan shutter_camera{1})]
[node playback{1} (start (20001 nil 22757.43))(duration 40)(type action)
        (plan (/ consolidate_tape{1} position_tape{3}))]
[node playback{2} (start (20170.29 nil 22926.71))(duration 73.28571)(type action)
        (plan (/ consolidate_tape{2} datarate_change_3))]
[node position_tape{1} (start (0 nil 7958.617))(duration 7.142857)(type action)
        (plan start_recording{1})]
                .
                .
                .
[node set_filter{1} (start (0 nil 2533.4))(duration 16.6)(type action)
        (plan shutter_camera{1})]
[node set_filter{2} (start (2405.76 nil 9176.5))(duration 23.5)(type action)
        (plan shutter_camera{2})]
[node shutter_camera{1} (start (2400 2500 2550))(duration 5.76)(type action)
        (plan (/ high_speed_slew set_filter{2} transmit_picture))]
[node shutter_camera{2} (start (9000 9000 9200))(duration 5.76)(type action)
        (plan record_picture)]
[node start_recording{1} (start (2948.76 nil 7965.76))(duration 1)(type action)
        (plan roll)]
[node start_recording{2} (start (9004.76 nil 9204.76))(duration 1)(type action)
        (plan record_picture)]
```

Figure 2: An abridged MIRRORS/II specification file for the self-processing network of the complete Spaceworld plan. Some details were omitted for brevity. (Page 2 of 3.)

```
[node transmit_picture (start (2405.76 nil 2555.76))(duration 144)(type event)
        (plan (/ change_data_mode{2} begin_earth_occultation))]
                .
                .
                .
[node turn_on_camera (start (0 nil 2520))(duration 30)(type action)
        (plan shutter_camera{1})]
[node turn_on_gyros (type action)]
[node turn_on_heaters (type action)]
[node turn_on_transmitter (start (0 nil 2255.76))(duration 300)(type action)
        (plan transmit_picture)]
[node viewing{1} (start (2400 nil 2550))(duration 0)(type inference)
        (plan shutter_camera{1})]
                .
                .
                .
[node datarate_change_1 (start (1000 nil 1000))(duration 300)(type event)
        (plan (/ viewing{1} datamode_ok_3 start_recording{1}))]
                .
                .
                .

;--- Control specification
[control ALTCONTROL]
[transcript plan]
[events (clamp)(show)]
;--- Start the nodes in the network which have no predecessor nodes.
[clamp (from 0 thru 1) (plan (&
        medium_speed_slew
        turn_on_camera
        set_filter{1}
        change_data_mode{1}
        turn_on_transmitter
        position_tape{1}))
        1.0]
[clamp (from 1000 thru 1001) (plan datarate_change_1) 1.0]
[run 25002]
[exit]
```

Figure 2: An abridged MIRRORS/ii specification file for the self-processing network of the complete Spaceworld plan. Some details were omitted for brevity. (Page 3 of 3.)

A marker-passing paradigm was used as the spreading activation method for this self-processing network. Basically, a node representing a goal in a plan passes a marker to the nodes to which it sends outgoing connections, representing goals in the plan which are dependent on the completion of the sending node, when it has finished executing. Once a node which has not executed yet has received markers from all the nodes from which it receives incoming connections thereby satisfying the dependency constraint, and the starting time constraint has been satisfied, the node can begin executing. Nodes which do not have any dependency constraints can begin executing as soon as their starting time constraints are satisfied.

Using the network specification shown in Figure 2 and the marker-passing method described above the Spaceworld plan executed successfully, taking advantage of the parallelism inherent in the plan. The results of this execution can be seen in Figure 3. The output is composed of messages generated by nodes; each node prints a message to indicate when it began executing and when it stopped executing. Figure 3 shows that goals in the plan which are independent of each other are executed in parallel while other goals which must be executed in a specific order are executed sequentially. For example the nodes turn_on_camera and turn_on_transmitter both begin executing at the same time indicating that they are not dependent on each other and can be executed in parallel. Also, observe by comparing the network specification in Figure 2 to the output in Figure 3 that the starting time constraints of each goal have been satisfied and the duration time of each goal is accurate.

```
Beginning simulation, will stop at iteration 25002
Starting action change_data_mode{1} at time 0.
Starting action medium_speed_slew at time 0.
Starting action position_tape{1} at time 0.
Starting action set_filter{1} at time 0.
Starting action turn_on_camera at time 0.
Starting action turn_on_transmitter at time 0.
  Finished action position_tape{1} at time 7.
  Finished action set_filter{1} at time 16.
  Finished action turn_on_camera at time 30.
  Finished action medium_speed_slew at time 80.
Starting event platform_damping_2 at time 80.
  Finished event platform_damping_2 at time 180.
  Finished action change_data_mode{1} at time 300.
  Finished action turn_on_transmitter at time 300.
Starting event datarate_change_1 at time 1000.
  Finished event datarate_change_1 at time 1300.
Starting inference viewing{1} at time 2400.
  Finished inference viewing{1} at time 2400.
Starting inference datamode_ok_3 at time 2405.
  Finished inference datamode_ok_3 at time 2405.
Starting action shutter_camera{1} at time 2500.
  Finished action shutter_camera{1} at time 2505.
Starting action high_speed_slew at time 2505.
Starting action set_filter{2} at time 2505.
Starting event transmit_picture at time 2505.
  Finished action set_filter{2} at time 2528.
  Finished action high_speed_slew at time 2604.
Starting event platform_damping_1 at time 2604.
  Finished event transmit_picture at time 2649.
Starting action change_data_mode{2} at time 2649.
  Finished event platform_damping_1 at time 2904.
Starting action start_recording{1} at time 2948.
  Finished action change_data_mode{2} at time 2949.
  Finished action start_recording{1} at time 2949.
Starting action roll at time 2949.
  Finished action roll at time 3879.
Starting inference viewing{2} at time 3879.
  Finished inference viewing{2} at time 3879.
Starting action change_data_mode{3} at time 3888.
  Finished action change_data_mode{3} at time 4188.
Starting event begin_earth_occultation at time 5000.
  Finished event begin_earth_occultation at time 5001.
Starting event datarate_change_2 at time 7000.
  Finished event datarate_change_2 at time 7300.
Starting action shutter_camera{2} at time 9000.
```

Figure 3: Output of the self-processing network for non-hierarchical plan execution. (Page 1 of 2.)

```
Starting action start_recording{2} at time 9004.
  Finished action shutter_camera{2} at time 9005.
  Finished action start_recording{2} at time 9005.
Starting event record_picture at time 9005.
  Finished event record_picture at time 9045.
Starting action change_data_mode{4} at time 9053.
Starting action turn_off_tape_recorder at time 9053.
  Finished action turn_off_tape_recorder at time 9054.
Starting action position_tape{2} at time 9054.
  Finished action position_tape{2} at time 9094.
  Finished action change_data_mode{4} at time 9353.
Starting event end_earth_occultation at time 20000.
  Finished event end_earth_occultation at time 20001.
Starting action playback{1} at time 20001.
  Finished action playback{1} at time 20041.
Starting action consolidate_tape{1} at time 20049.
Starting action position_tape{3} at time 20049.
  Finished action consolidate_tape{1} at time 20049.
  Finished action position_tape{3} at time 20170.
Starting action playback{2} at time 20170.
  Finished action playback{2} at time 20243.
Starting action consolidate_tape{2} at time 20243.
  Finished action consolidate_tape{2} at time 20243.
Starting event datarate_change_3 at time 23000.
  Finished event datarate_change_3 at time 23300.
Starting inference clear_tape at time 25000.
  Finished inference clear_tape at time 25000.
```

Figure 3: Output of the self-processing network for non-hierarchical plan execution. (Page 2 of 2.)

V. Hierarchical Plan Execution

While non-hierarchical plans like that considered above order goals at a single level of abstraction, hierarchical plans consist of multiple levels of abstraction where each level "deeper" in the plan represents a more detailed level of abstraction. Based on the "lattice controller" described in [7] we were inspired to extend our research to include hierarchical plan execution. Hierarchical plans are generated by some AI planning systems so any general purpose plan execution scheme must be able to handle them. Hierarchical plans avoid some aspects of the computational complexity arising in real-world applications and are therefore of great value.

To develop a hierarchical plan to use for this research we added higher levels of abstraction to a subset of Vere's Spaceworld plan. The resulting plan can be seen in Figure 4. Each higher-level goal in the plan can be decomposed into more detailed goals. For example, the highest level goal in the plan is "Take a picture of the satellite Clotho" (take_picture_clotho). This goal can be broken down into the more detailed goals of "Photograph the satellite" (photograph_satellite) and "Transport the picture to earth" (transmit_picture_to_earth). Further levels of abstraction are illustrated in Figure 4. Goals in Figure 4 which have a numeric duration time are goals from the original non-hierarchical plan. The goals added to form the hierarchical plan do not have a specified duration time since their duration time is dependent on the starting and duration times of the goals one level lower in the hierarchy. Starting times of the higher-level goals were calculated based on the *earliest* early starting time and *earliest* latest starting time of goals of which the higher level goals are composed. For example, the early starting times of the nodes which are "children" of node photograph_satellite in the hierarchy are 0, 0, 0, 0, and 2400 so the early starting time of photograph_satellite node is 0 and the latest starting times of the children nodes are 2520, 2520, 2370, 1000, and 2550 so the latest starting time of the photograph_satellite node is 1000.

Nodes in the hierarchical planning network have two new node attributes, *parent* and *depend*, in addition to the start and duration attributes used in the non-hierarchical plan network. The parent attribute

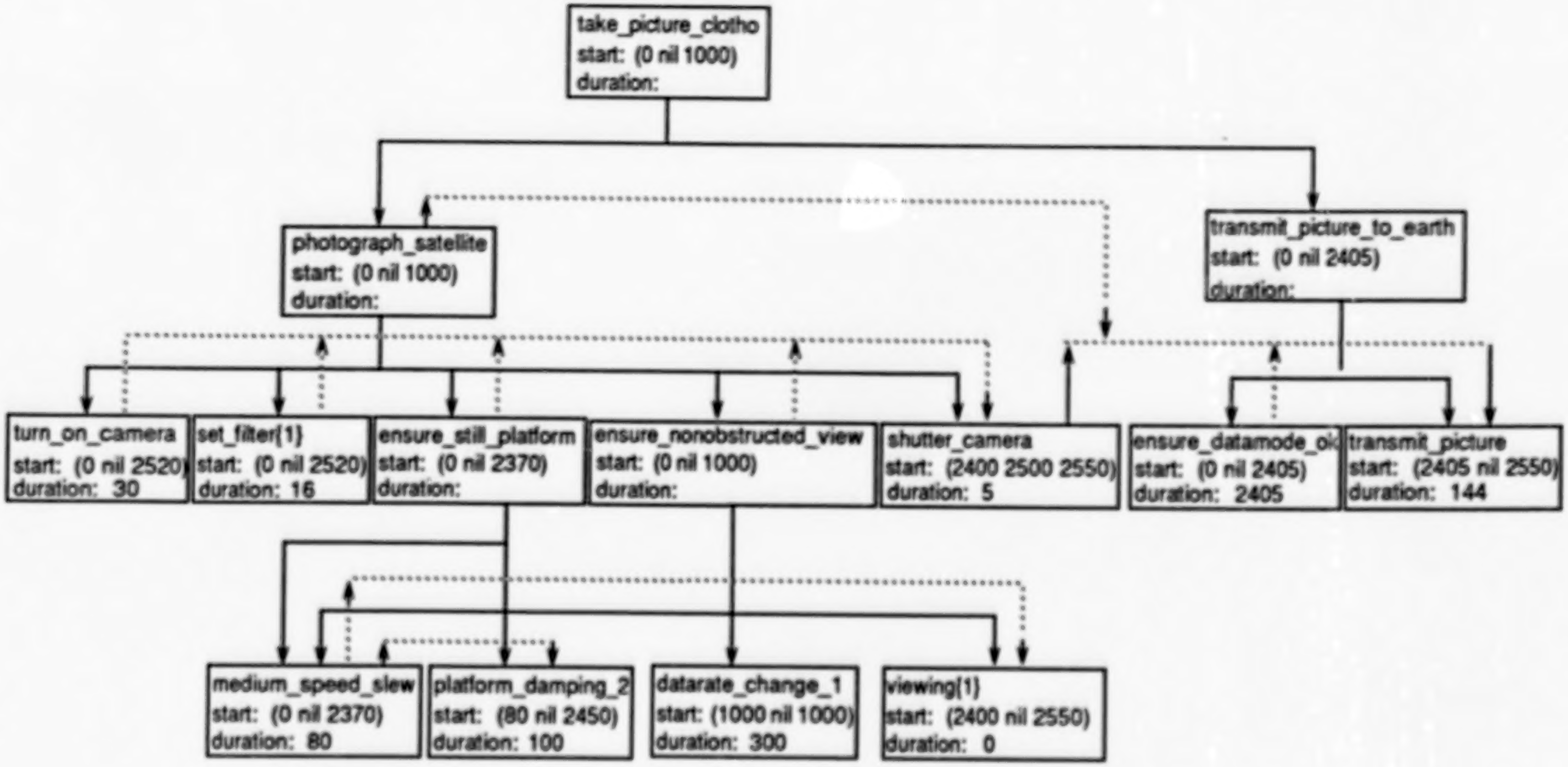

take_picture_clotho
start: (0 nil 1000)
duration:
photograph_satellite
start: (0 nil 1000)
duration:
transmit_picture_to_earth
start: (0 nil 2405)
duration:
turn_on_camera
start: (0 nil 2520)
duration: 30
set_filter{1}
start: (0 nil 2520)
duration: 16
ensure_still_platform
start: (0 nil 2370)
duration:
ensure_nonobstructed_view
start: (0 nil 1000)
duration:
shutter_camera
start: (2400 2500 2550)
duration: 5
ensure_datamode_ok
start: (0 nil 2405)
duration: 2405
transmit_picture
start: (2405 nil 2550)
duration: 144
medium_speed_slew
start: (0 nil 2370)
duration: 80
platform_damping_2
start: (80 nil 2450)
duration: 100
datarate_change_1
start: (1000 nil 1000)
duration: 300
viewing{1}
start: (2400 nil 2550)
duration: 0

Figure 4: A hierarchical plan based on a portion of the Spaceworld Plan.

indicates what node or nodes are "above" a given node in the plan hierarchy (the solid lines in Figure 4). The depend attribute indicates nodes in the hierarchy which depend on the given node's execution completion to begin their own execution (the dotted lines in Figure 4). This order-dependent information is represented by node connections in both the non-hierarchical and hierarchical plan network. However, in the hierarchical plan, network node connections are also used to indicate more detailed goals which combine to represent the "parent" goal node. For example, the take_picture_clotho node connects to the photograph_satellite and transmit_picture_to_earth nodes since the latter two nodes represent a finer level of abstraction of the "parent" goal node take_picture_clotho (see Figure 5).

```
;--- Set containing the goals (nodes) in a hierarchical sebset of a Spaceworld plan.

[set plan (method hplan)                    ;--- Activation method for hierarchical plans.
        [contains
                take_picture_clotho photograph_satellite
                transmit_picture_to_earth turn_on_camera set_filter{1}
                ensure_still_platform ensure_nonobstructed_view
                shutter_camera{1} ensure_datamode_ok transmit_picture
                medium_speed_slew platform_damping_2 datarate_change_1 viewing{1})
        (attribute start dynamic)                       ;--- Definition of node attributes needed for this model.
        (attribute duration dynamic optional)
        (attribute parent dynamic optional)
        (attribute depend dynamic optional)
        (attribute type dynamic)
        (connects (plan oto incoming optional))]   ;--- Nodes in this set connect to other nodes in this set.

;--- The following node statements give detailed node attribute information
;--- for each node as well as describing the node connections in the network.

[implicit (member plan)]

[node take_picture_clotho (type action)(start (0 nil 1000))
        (plan (/ photograph_satellite transmit_picture_to_earth))]

[implicit (parent (take_picture_clotho))]
[node photograph_satellite (type action)(start (0 nil 1000))
        (depend (transmit_picture))
        (plan (/ turn_on_camera set_filter{1} ensure_still_platform
                ensure_nonobstructed_view shutter_camera{1} transmit_picture))]
[node transmit_picture_to_earth (type action)(start (0 nil 2405))
        (plan (/ ensure_datamode_ok transmit_picture))]

[implicit (parent (photograph_satellite))]
[node turn_on_camera (type action)(start (0 nil 2520))(duration 30)
        (depend (shutter_camera{1}))
        (plan shutter_camera{1})]
[node set_filter{1} (type action)(start (0 nil 2533))(duration 16)
        (depend (shutter_camera{1}))
        (plan shutter_camera{1})]
[node ensure_still_platform (type action)(start (0 nil 2370))
        (depend (shutter_camera{1}))
        (plan (/ medium_speed_slew platform_damping_2 shutter_camera{1}))]
```

Figure 5: A MIRRORS/II specification file for the self-processing network of the hierarchical plan shown in Figure 4. (Page 1 of 2.)

```
[node ensure_nonobstructed_view (type action)(start (0 nil 1000))
        (depend (shutter_camera{1}))
        (plan (/ medium_speed_slew datarate_change_1 viewing{1} shutter_camera{1}))]
[node shutter_camera{1} (type action)(start (2400 2500 2550))(duration 5)
        (depend (transmit_picture))
        (plan transmit_picture)]

[implicit (parent (transmit_picture_to_earth))]
[node ensure_datamode_ok (type action)(start (0 nil 2405))(duration 2405)
        (depend (transmit_picture))
        (plan transmit_picture)]
[node transmit_picture (type action)(start (2405 nil 2555))(duration 144)]

[node medium_speed_slew (type action) (start (0 nil 2370))(duration 80)
        (parent (ensure_still_platform ensure_nonobstructed_view))
        (depend (platform_damping_2 viewing{1}))
        (plan (/ platform_damping_2 viewing{1}))]
[node platform_damping_2 (type action)
        (start (80 nil 2450))(duration 100)
        (parent (ensure_still_platform))]

[implicit (parent (ensure_nonobstructed_view))]
[node datarate_change_1 (type event)(start (1000 nil 1000))(duration 300)
        (depend (viewing{1}))(plan viewing{1})]
[node viewing{1} (type inference)(start (2400 nil 2550))(duration 0)]

;--------- control spec
[events (input)(show)]
[control ALTCONTROL]
;--- Start the top-level node in the hierarchy
[input (from 0 thru 1)(plan take_picture_clotho) 1.0]
[transcript hplan]
[run 2655]
[exit]
```

Figure 5: A MIRRORS/II specification file for the self-processing network of the hierarchical plan shown in Figure 4. (Page 2 of 2.)

The marker passing algorithm used for hierarchical plan execution differs slightly from the one used for non-hierarchical plan execution. In hierarchical plan execution, goal execution begins with the node at the top-most level of abstraction. Once a parent node begins executing it sends a marker to each of its "child" nodes. Each "child" node must receive a marker from each node on which it depends and from it's parent node before it can begin executing; it is not appropriate to begin executing nodes at lower levels of the hierarchy if the conditions for executing their parent goals at higher levels of the hierarchy have not been met. Once a "child" node has completed executing, it sends a marker to its "parent" node indicating that it is done and to any other nodes which depend on the completion of its execution. All the child nodes (those one level lower in the hierarchy) must complete executing before the parent node's execution can be considered complete. The starting time constraints remain the same.

Using the self-processing network shown in Figure 5 and the marker-passing method described above, the hierarchical plan executed successfully. The results are shown in Figure 6. You can see that the top-most node began executing first and finished executing last because it could not complete executing until all its lower-level detail nodes had finished executing. Many of the goals were executed in parallel. Also note that the necessary sequential processing was maintained. For example, the transmit_picture goal did not begin executing until the photograph_satellite node was finished executing.

```
   Beginning simulation, will stop at iteration 2655
   Starting action take_picture_clotho at time 0.
   Starting action photograph_satellite at time 0.
   Starting action transmit_picture_to_earth at time 0.
   Starting action turn_on_camera at time 0.
   Starting action set_filter(1) at time 0.
   Starting action ensure_still_platform at time 0.
   Starting action ensure_nonobstructed_view at time 0.
   Starting action ensure_datamode_ok at time 0.
   Starting action medium_speed_slew at time 0.
     Finished action set_filter(1) at time 16.

  Finished action turn_on_camera at time 30.
  Finished action medium_speed_slew at time 80.
Starting action platform_damping_2 at time 80.
  Finished action platform_damping_2 at time 180.
  Finished action ensure_still_platform at time 191.
Starting event datarate_change_1 at time 1000.
  Finished event datarate_change_1 at time 1300.
Starting inference viewing(1) at time 2400.
  Finished inference viewing(1) at time 2400.
  Finished action ensure_nonobstructed_view at time 2402.
  Finished action ensure_datamode_ok at time 2405.
Starting action shutter_camera(1) at time 2500.
  Finished action shutter_camera(1) at time 2505.
  Finished action photograph_satellite at time 2506.
Starting action transmit_picture at time 2506.
  Finished action transmit_picture at time 2650.
  Finished action transmit_picture_to_earth at time 2651.
  Finished action take_picture_clotho at time 2652.
```

Figure 6: Output of the self-processing network for hierarchical plan execution.

VI. Discussion

Along with recent related work [7], this study demonstrates for the first time that plan execution, a task usually solved using traditional AI problem-solving techniques, can be accomplished using a self-processing network. The distributed processing approach used here allows many plan steps to be executed in parallel while still preserving the essential sequential aspects of the plan execution. The fact that self-processing networks have been applied to various space-related applications [5,6] in addition to the one discussed here demonstrates the general aplicability of this approach to planning and control problems relevant to spacecraft activities. A logical next step might be to implement a self-processing model which could execute plans which are dynamically changing during the period of execution.

This work also demonstrates that MIRRORS/II is a powerful environment for the development/evaluation of self-processing systems in general. It allowed us to develop this plan execution model in a very short amount of time and to implement marker passing processing paradigms as needed. The design of MIRRORS/II and all previous work with MIRRORS/II had been limited to connectionist models and had not considered the possibility of using methods like marker passing. The ease with which MIRRORS/II supported marker passing methods without any alterations to MIRRORS/II itself suggests that it will prove quite robust as a software environment for future automation research. A logical next step might be the development of other parallel AI methods in the context of MIRRORS/II.

Acknowledgements: Supported in part by NASA award NAG1-885 and in part by NSF award IRI-8451430.

VII. References

[1] D'Autrechy, C.L., et al., A General-Purpose Environment for Developing Connectionist Models, *Simulation*, 51, 1988, 5-19.

[2] D'Autrechy, C.L., et al., *MIRRORS/II Reference Manual*, 1988.

[3] Reggia, J. & Sutton, G., Self-Processing Networks and Their Biomedical Implications, *Proc. of the IEEE*, 76, 1988, 680-692.

[4] Vere, S., Planning in Time: Windows and Durations for Activities and Goals. *IEEE Trans. Pat. Anal. & Mach. Intel.*, 1988.

[5] Whitfield, K., et al., A Competition-Based Connectionist Model for Dynamic Control, in this Proceedings, 1989.

[6] Peng, Y. and Reggia, J., A Connectionist Model for Diagnostic Problem Solving, *IEEE Trans. Sys., Man and Cyber.*, 1989, in press.

[7] Sliwa, N. and Soloway, D., A Lattice Controller for Telerobotic Systems, *American Controls Conference*, 1987.

BLANK PAGE

MISSION SCHEDULING

Christine GASPIN

Centre d'Etudes et de Recherches de Toulouse
Groupement d'Intelligence Artificielle
2 avenue Edouard Belin 31 055 Toulouse Cedex
Tel: 61 55 71 11 poste 7442
Telex: 521 596 F onecert

Abstract

We investigate through a mission scheduling problem how a neural network can work compared to a hybrid system based on operation research and artificial intelligence approach. Then we present a discussion to demonstrate the characteristic features of each one.

Keywords

Scheduling, assignment, operation research, heuristics, optimization, neural network, self-organization

1 Introduction

Problems that have combinatorial complexity are generally said NP-complete problems (meaning that there are no polynomial algorithm for finding optimal solutions). About solving combinatorial optimization problems, standard operation research methods are effective for little problems but they often failed when more complex problems involving many variables and constraints are to be solved.

Whenever we cannot use standard methods, more suitable "hybrid" systems merging operation research and artificial intelligence techniques but very domain dependant and less restricting, work with specific heuristics and offer non optimal but very satisfying solutions.

A more general "hybrid" system, OSCAR, based on an automatic intelligent reasoning has been built in [9] for the mission scheduling problem, using a general assignment algorithm designed to work with a variety of heuristics and rules to make choices and define the reasoning strategies.

NP-complete problems also have been shown to be solved by a neural network approach if they can be formulated as optimization problems [3]. Indeed many researchers have shown that neural networks are satisfying constraint systems and

that we are able to design neural networks giving very good solutions for operation research problems.
A number of papers have been yield which compare different neural network techniques for solving optimization problems [5,6,7].

Whenever it is known [3] that computational power and speed of collective analog networks of neurons in rapidly solving optimization problems has been demonstrated, we ask what are really interests of a neural approach compared to other approaches through a specific assignment problem.

When OSCAR works with local measures, we propose to compare, for the same assignment problem, such a system to an optimization method based on neural networks involving a global measure.

After briefly presenting the assignment problem, we give the principle of the general assignment algorithm then the neural network approach. Then we discuss about properties of both approaches giving our topics about weakness and strength of each one.

2 The scheduling problem

Interests

A spacecraft scheduling is a difficult problem because of a large number of different and interacting constraints, uncertainty and often situation-dependant optimization criteria which make the search process computationaly complex.

The mission scheduling problem consists of finding both a set of resources and a temporal position for each elementary lower-level activity. Our interest deals with three points:

- Temporal relative and absolute constraints

- Activities which are elementary action to be performed on the spacecraft or on the ground requiring one resource-group to be chosen among various possible resource-groups.

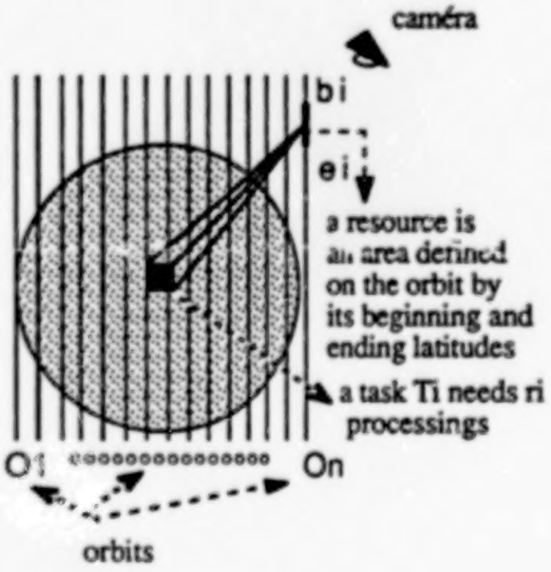
- Pointwise unsharable resources defined in [9] as resources involving both the assignment and scheduling problems. Such a resource is for example a camera and the camera assignment problem has to be the following.

The camera "resource"

An heliosynchronous spacecraft payload consists of cameras with tilting capabilities. Therefore, its workload consists of activities A_i which are processed by taking r_i pictures of a given area from a possible set of specified orbits. The area is defined on the orbits by its beginning (b_i) and ending (e_i) latitudes and has a given width. Mirrors allow to take pictures of the landbelt not only just below the satellite but also above several landbelts beside the central band.

Since weather conditions can alter the quality of a picture, r_i pictures are required to satisfy an activity i.e. get a good picture of all the expected areas.

Let r_i the required number of processings of activity A_i. The workload is built dynamically and thus the assignment of the activities must be updated after each request. The sequence of orbits is given and cannot be changed in any way. Therefore at a given time t, the following orbits $O_1 \ldots O_n$ are described with the activities which are assigned to.


caméra
bi
ei
a resource is an area defined on the orbit by its beginning and ending latitudes
a task Ti needs ri processings
O1
On
orbits

Why this is a "conflict-solving" system is that in this activity assignment problem we have to make choices that is to find a resource (an orbit interval) among several possible unsharable resources respecting constraints.

3 A "hybrid" system: OSCAR

OSCAR merges operation research and artificial intelligence methods for solving a large class of spatial missions assignment problems (more precisely scheduling problems).

The set of problems OSCAR is able to solve is the following :

given a set of activities, the goal is to find both a set of resources and a temporal position for each elementary activity to process.

Let us define:

- An **activity**
cannot be split and has to be assigned on the same resource-group. Several resource-groups are possible choices for an activity

- A **resource-group**
is defined with

. a set of physical resources
. a time window
. a duration
. temporal constraints

- **Physical resources**
are to be sharable or unsharable, consummable (memory) or not (electrical power, camera).

- **Temporal constraints**
are **absolute** (beginning and ending times of an activity are bound to stay within a given interval) or **relative** (between two activities).

Therefore the goal is to find for each activity

. a resource-group
. an exact temporal position

This approach is **incremental** in that an activity enters one after another one from the set of the all activities in a growing context. The current context is a view of already assigned activities but it is possible to put this assignment into question. The philosophy is the following:

Let A be an activity to be entered in the context

- If the current context has enough room for A, one of the possible resource-groups is assigned to A.

- Otherwise, the mimimal sets of activities that prevent A from being assigned one resource-group are defined. Then a minimal set of activities is chosen, ejected from the current context assigning a group-resource to A, and has to be re-planned.

- If one of the ejected activities cannot be re-entered into the context, another minimal set is chosen: there is a failure

- If all the minimal sets fail, it is the total failure

- If all the ejected activities are to be re-planned, the process succeeds

It has been proved that the general assignment algorithm has the following properties :

- It **terminates**.

- It is **complete**: if a solution exits, it finds it.

- Theoretical and pratical studies about the average computational **complexity** of the algorithm have shown that except for few very constrained cases, the average number of activities that have to be re-planned doesn't depend on the number of activities but only depends on the saturation rate of the context (rate activities /possible resources).

Specific and general heuristics are used for choosing:

- A group-resource in the context

- A minimal set of activities

OSCAR has been successfully tested on camera resource assignment problem concerning an Earth-Obsevation Satellite.

4 The neural network approach

We are interested here in solving a complex optimization problem in parallel without learning process. About solving constraint satisfaction problems [10] gives three neural networks models :

- The Hopfield network [1].

- The Boltzmann machine [4].

- The Tank and Hopfield network [3].

Our goal is not here to compare the three models applied to a specific problem (we find comparisons of models through the travelling salesman and the graph bisection problems in [5,7]) but we want to show how they efficiently can help to solve an assignment problem, what are their weakness and strength compared with the previous approach.

We use both Hopfield networks and Boltzmann machines during our simulations.
These models consist of a large number of computing elements called units that are connected to each other by bidirectional links and this massive interconnection gives them an important computational power.

With each unit **u_i** a binary value is associated, denoting its state "0" or "1," (on or off). Therefore, at a given time the network can be represented by a state vector. Weights on links are symetric, (having the same strength in both directions). A solution is given by a configuration C of the network that is the state vector.
An "objective" function of a global configuration is defined by analogy with statistical mechanics energy

$$E = -1/2\, \Sigma_i\, \Sigma_{j\neq i}\, w_{ij}\, s_i s_j - \Sigma_i I_i s_i$$

where

w_{ij} is the strength of connections between units u_i and u_j,

s_i is the state of unit i,

I_i is the threshold of unit i.

If the units change their states one at a time (0 ---> 1), given a configuration C, then firstly a neighbouring configuration C' is generated, changing unit u_i state and secondly it is evaluated.

Because the connections are symetric, the difference between the energy of configuration C and C' can be determined locally by the unit u_i and

$$\Delta E_i = E_{si=0} - E_{si=1} = \Sigma_k\, w_{ki}\, s_k + I_i$$

thus allowing a parallel execution.

In an **Hopfield model** the rule decision is :

$$s_i = \begin{array}{l} 0 \text{ if } \Delta E_i < I_i \\ 1 \text{ otherwise} \end{array}$$

In the Boltzmann machine, the units are in one of the two states determined as a probabilistic function of the states of its neighboring units (these which are connected to) and the weights on its links to them. The acceptance probability in changing a unit state is

$$p(s_i = 1) = 1 / (1 + \exp(-\Delta E_i/T))$$

where

T is the computational temperature used in simulations as a control parameter associated with a simulated annealing process which is a statistical generalization of hill-climbig optimization methods. It allows to make uphill moves instead of settling in local minimum.

T=0 is the limit-case used in the Hopfield model.

The dynamics of evolution of these systems states follow a simple rule (above) and is asynchronous (a unit is randomly chosen and tries to change its state given its inputs).

The updating rule of a unit state is an energy optimizing rule (minimizing / maximizing). Modifications of units states continue until a stable state is reached, that is, an energy optimum is reached.

Mapping a satisfaction constraints problem

For mapping a satisfaction constraint problem we have to:

- **Find a representation of the problem** as to be solved by a neural network.

Indeed, the litterature often involve schemes which are operation research ones (matrix, graphs) and we have to investigate if it is relevant to relate operation research representation problems to neural techniques.

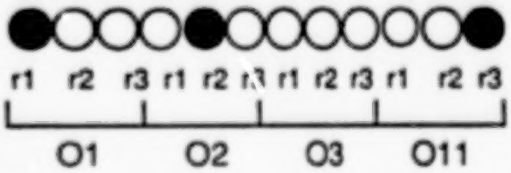
The solution to the assignment problem consists of an optimal assignment with the respect to the interacting constraints.

To map this problem onto the computational network, we require a representation that needs to be decoded. We have chosen a representation scheme in which the final assignment is specified by a configuration of the network.

For example, if an activity Ai requires ri = 3 processings and if a possible set of resources for Ai is

Qi = {O1(1 10) O2(3 13) O3(0 9) O11(10 18)}

a possible solution is given by



- The processing r1 of Ai is on the orbit O1(1 10)

- The processing r2 of Ai is on the orbit O2(3 13)

- The processing r3 of Ai is on the orbit O11(10 18)

- **Define units.** Therefore a unit is the elementary information to process in this optimization problem and it means :

" a possible choice " for assigning the activity A_i to the resource O_j respecting its temporal constraints.

This representation scheme is natural since any activity can be assigned to any one in its related possible set of resources.

The number of units required for a problem where p activities are involved is therefore

$$N = \Sigma_p n_p \text{ units}$$

where

n_p

=

(the number of processings of activity p)

*

(the number of possible orbits intervals which can be assigned to activity p)

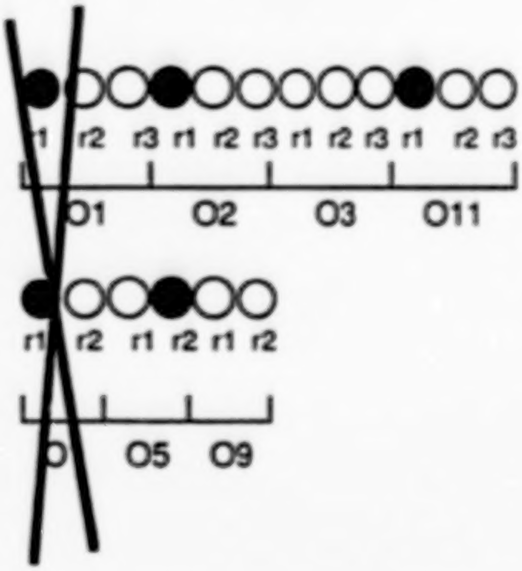
that is the number of units for representing the activity A_p.

Our point is that resources are unsharable that is

- any part of any orbit cannot be simultaneously shared between two activities. If $A_{i'}$ is another activity and Qi' its possible set of resources and $r_{i'}=2$

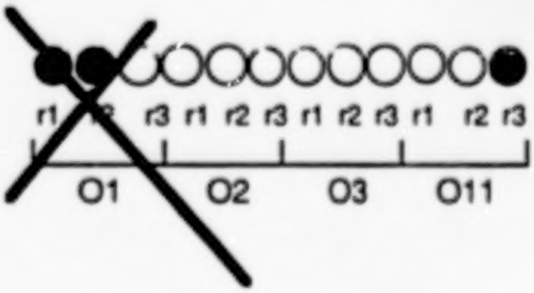
$Q_{i'}$= {O1(1 10) O5(5 14) O9(7 16)}

we cannot find the following solution scheme

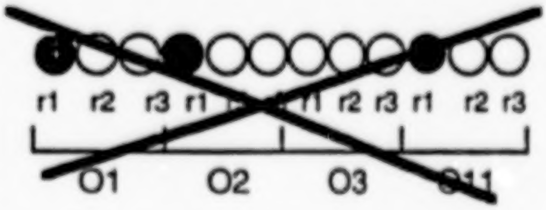

r1 r2 r3 r1 r2 r3 r1 r2 r3 r1 r2 r3
O1 O2 O3 O11
r1 r2 r1 r2 r1 r2
O O5 O9

that is we cannot find r1 on orbit O1 for activity A_i and r2 on orbit O1 for activity $A_{i'}$.

- different processings of the same activity cannot share the same part of an orbit. So we cannot find the following solution scheme

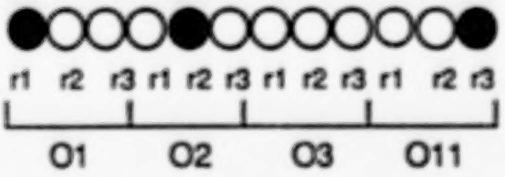

r1 r3 r1 r2 r3 r1 r2 r3 r1 r2 r3
O1 O2 O3 O11

- the same processing cannot be assigned more than one resource at a time. So we cannot find


r1 r2 r3 r1 r1 r2 r3 r1 r2 r3
O1 O2 O3 O11

that is r1 cannot be assigned to orbits O1 and O2

- an activity try to assign all the processing it requires:



The "objective" is to occupe as better as possible the all set of resources.

Because of these choices a unit u_{ijk} means the possibility for activity i to assign processing j on orbit k

- Define the connections that is define what a link is with the respect to the constraints and the "objective" of the problem.

Weights on links represent a weak pairwise constraint between two hypotheses.

i : index giving the orbits
j : index giving the realizations
t : index giving the tasks

Connections are built between links satisfying the following constraints

Inhibitory links

.Two processing of the same task do not share a part of the same orbit.

.Two processing of different tasks do not share a part of any orbit.

. A task Ti can't plan more than ri processing

Excitatory links

. Each unit activates a processing of another orbit

- Define what are the right connection weights

For computing a solution to the problem, the network has to be described by an energy function in which the most stable state of the network is the best assignment. Our energy function is the following

$$E = -a\Sigma_{i,i'\in c1}(r_i{*}(p_i+p_{i'}))s_is_{i'}$$

$$-b\Sigma_{i,i'\in c2}(\max(p_i,p_{i'}))s_is_{i'}$$

$$-c\Sigma_{i,i'\in c3}(\min(p_i+p_{i'}))s_is_{i'}$$

$$+d\Sigma_{i,i'\in c4}(r_i{*}(p_i+p_{i'}))s_is_{i'}$$

where

$c1=\{u_i,u_{i'}/$ activity$_i$ = activity$_{i'}$
orbit$_i$ = orbit$_{i'}$
processing$_i\neq$processing$_{i'}\}$

$c2=\{u_i,ui'/$ activity$_i$ = activity$_{i'}$
orbit$_i$ ≠ orbit$_{i'}$
processing$_i$= processing$_{i'}\}$

$c3=\{u_i,u_{i'}/\ \text{activity}_i \neq \text{activity}_{i'}$
$\text{orbit}_i = \text{orbit}_{i'}$
$(\text{processing}_i \neq \text{processing}_{i'})$
or
$(\text{processing}_i = \text{processing}_{i'})\}$

$c4=\{u_i,u_{i'}/\ \text{activity}_i = \text{activity}_{i'}$
$\text{orbit}_i = \text{orbit}_{i'}$
$\text{processing}_i = ((\text{processing}_{i'}+1)$
$\text{mod (number of proceesing}_i))\}$

$a,b,c,d > 0$ are parameters

$p_i =$ F(number of possible resources for A_i, number of activities for which orbit related to u_i is a possible resource, number of processings needed by A_i)

This function gives advantage to the less requested resources.

5 Starting a discussion

- Complexity

Complexity is defined in computer science by :

- Spatial complexity
- Temporal complexity

The neural network simulations always converged giving satisfying solutions from an engineering point of view for little examples with 256 units (25 activities and 21 resources) or less.

If the complexity has been theoretically and pratically studied in the R.O. and I.A. approach giving very interesting results about the average computational complexity of the algorithm, for very constrained cases, combinatorial problems are again involved.

Theoretical studies are being made for the neural net one. Nevertheless, in this application the neural network appears to converge with the number of units.

If many activities are involved in a problem, a great number of possibilities appears. Therefore, solving such a complex optimization problem needs a large number of units and a highly connected network. But because of the 10^{11} neurons and 10^6 connections from each one in the nervous system and very encouraging results which appear in the domain, it is relevant to investigate how the computational power of these networks can help to solve optimization problems.

Morever, neural networks are parallel systems which are functionnally implementable .

- Locality of taking decisions

The assignment algorithm decides with locality. Indeed, when a new task enters, only its possibility of assignment is investigated.
The new task is or is not able to be planned. No global criteria is taken into account. An already planned task is ejected only if it is sure to get another assignment.

A neural net approach also decides locally about an assignment. Indeed the model is based on computational locality of decision in the units. Morever, it takes immediatly into account the constraints on other units that is the constraints of the problem. Therefore, there is a "microscopic" decision system (cooperation /competition) in every unit because of the neighbourood.(defined by the connections.)

- Global / Incremental decisions

The set of constraints mapped on the connections to a unit makes the unit to take locally a global decision that is it computes a global criteria to optimize.

We give for example in [8] a criteria to maximize "the probability of satisfying all the activities". Therefore an already planned task is ejected if the new activity optimizes the "objective" function satisfying the constraints given by the neighbours (the units which it is directly connected to).

In comparison, the research operation and artificial intelligence approach is an incremental decision process in which no global criteria is computed.

If the objective is to use all the ressources realizing the best assignment we find the following results for each approach:

Example

Activities	1	2	3	4
number of processings	2	2	2	2
lattitude of begin	1	1	1	1
lattitude of end	6	6	6	6
list of orbits	o1	o3	o3	o4
	o2	o4	o6	o5
	o3		o1	

In this example the R.O. and I.A. approach can only cope with two activities:

-T1 enters and is satisfied with T1 on orbits O1 and O2
or
T1 on orbits O1 and O3
or
T1 on orbits O2 and O3.

-T2 enters and is satisfied with T2 on orbits O3 and O4, and T1 on orbits O1 and O2.

-T3 cannot be satisfied without definitively ejecting T1 or T2

-T4 cannot be satisfied without ejecting T2

In that case, our network satisfied T1, T3 and T4, i.e. three tasks:

-T1 on orbits O1 and O2
or
T1on orbits O1 and O3
or
T1 on orbits O2 and O3

-T3 on orbits O1 and O6
and T1 on orbits O2 and O3
or
T3 on orbits O3 and O6
and T1 on orbits O1 and O2.

-T4 on orbits O4 and O5
and T3 on orbits O1 and O6
and T1 on orbits O2 and O3
or
T4 on orbits O4 and O5
and T3 on orbits O3 and O6
and T1 on orbits O2 and O3.

- Programming complexity

In the neural network formulation of an optimization problem, the constraints and the criteria to optimize are expressed inside the objective function. Thus by optimizing the objective function we optimize the satisfaction of the constraints. Morever, this approach provides an efficient and very simple technique for mapping the constraints on the links between units.

In the R.O. and I.A. approach, constraints on resources and activities are these which tend to reduce the combinatoric aspect and that is why the constraint-base reasoning is a more complex propagation mecanism.

6 Conclusion

We have described here how an optimization problem can be rapidly and easily mapped on a neural network and provide encouraging results in comparison with an operation research and intelligence artificial approach.

Nevertheless for neural networks which deal with such complex combinatorial optimization problems, the difficult task consists of finding weights of the connections. Indeed, because the solutions to these problems are not known as in other problems involving learning algorithms [6], they must be established by the designer.

Our interest is now to design a mapping method for solving constraint satisfaction problems (more precisely scheduling problems) through applications exploration and theoretical results.

7 References

[1] HOPFIELD, J.J., *Neural Networks and Physical Systems with Emergent Collective Computational Abilities*, Proc. Nat. Academy Sciences, USA, 79 (1982)2554

[2] S. KIKPATRICK, C. D. GELATT Jr.,M.P. VECCHI, *Optimization by Simulated Annealing*, Science(220),1983, p.671

[3] HOPFIELD, J.J. and D.W. TANK, *Neural Computation of Decision in Optimization Problems*, BiologicalCybernetics, 52(1985) 141

[4] J. McCLELAND, D.E. RUMELHART and al., *Parallel Distibuted Processing*, vol. 1 and 2 The MIT PRESS (1986)

[5] J.R. ANDERSON and C.PETERSON *Neural Networks and NP-Complete Optimization Problems. A performance Study on the Graph Bisection Problem*, MCC Technical report Number: EI-287-87, December 14, 1987

[6] Emile H.L. Aarts, Jan H.M. KORST, *Boltzmann Machines and their Applications*, Proc. of Parallel Architectures and Language, Europe Conf., June 1987, Eindhoven, The Netherlands

[7] James H. CERVANTES and Richard R. HILDEBRANT *Comparison of Three Neuron-Based Computation Schemes* Proc. of IEEE First Intern. Conf. on Neural Networks, Vol.III p. 657, SanDiego, California,June 21-24, 1987

[8] Ch. GASPIN, P. BOURRET and M. SAMUELIDES, *Neural etworks for optimal planning of a camera on board a satellite*, Proc. of Neuro-Nîmes, Nov.16-17 , 1988

[9] C. BADIE, G. VERFAILLIE *OSCAR : A hybrid system for mission scheduling, A.I. Applications for Space Projects*, European Space Agency ESTEC Noordwijk The Netherlands, November 15-16-17 1988

[10] Mark DERTHICK and Joe TEBELSKIS *"Ensemble" Boltzmann Units have Collective Computational Properties like those of Hopfield and Tank Neurons*, Neural Information Proc. Systems, Dana Z. Anderson Editor, American Institute of Physics, New York

On the Development of a Reactive Sensor-Based Robotic System

Henry H. Hexmoor
William E. Underwood, Jr.

AI Atlanta, Inc.
119 East Court Square
Decatur, GA 30030
(404) 373-7515

ABSTRACT

Flexible robotic systems for the space applications need to use local information to guide their action in uncertain environments where the state of the environment and even the goals may change. They have to be tolerant of unexpected events and robust enough to carry their task to completion. Tactical goals should be modified while maintaining strategic goals. Furthermore, reactive robotic systems need to have a broader view of their environments than sensory-based systems. We offer an architecture and a theory of representation extending the basic cycles of action and perception. This scheme allows for dynamic description of the environment and determining purposive and timely action. We are exploring applications of our scheme for assembly and repair tasks using a Universal Machine Intelligence RTX robot, but the ideas are extendable to other domains. This paper describes the nature of reactivity for sensor-based robotic systems and implementation issues we have encountered in developing a prototype.

1. Introduction

Almost every system in our daily life is reactive. They asynchronously accept input and produce output. This output may be normal or abnormal according to the intended function of the system. When they are situated in an environment, reactive systems produce responses based on their intended function and they do this by changing their goals and actions in response to new recognitions in the environment or shifting situations. Similarly, reactive programs form response behaviors based on "purposes" of the system within which they are embedded in response to asynchronous input from the environment. An operating system is a reactive system, whereas a program that blindly follows a set of instructions is not. Reactive systems developed to this date are aimed at producing routine, almost reflexive, behavior in an environment. Span of cognition and perception in these systems is narrow and often does not include an examination of "mental" states such as changes in goal priorities and long range expectations. We call this sort of reaction low-level reactivity and contrast it with high-level reactivity concerned with changing goals and reactive planning. Low-level reactivity might be suitable for small scale agents that are resource rich and have relatively low impact on the overall activities. On the other hand, intelligent agents slated for space applications have limited resources and need to react not only in the local sense of the word but also in a rational manner by building or altering goals or their specifications. This is required for reasoning in a dynamic problem space.

Most everyday activities are immediate and a myopic view of the world suffices to construct models of engagement in the world with no internal representation of the world. We argue that one needs to plan across activities as well as about them and this calls for a model of an agent interacting with a changing environment capable of adapting its behavior and reasoning at various levels of abstraction to purposively react. Purposive systems perform actions in relation to their functional requirements [Kim, 1988]. Reactive behavior is needed for improving use of resources for goal achievement. This model considers directed perception and interruptible cognitive processing in the perception-action cycle. Planning and reflective capabilities are crucial to robust reasoning agents.

Planning systems can be augmented with levels of abstraction in order to cope with combinatorics of detailed robot activitity. By carefully limiting the scope of planning to levels of abstraction, it becomes easier to identify stereotypical circumstances. A characteristic of reactive systems is hierarchies of activity. This hierarchy is a toolbox metaphor of behavioral skills. At one extreme the behavior may call for a dexterous manipulator with much sensory information, and at the other extreme it may provide a sufficing jesticulation behavior with minimal resources. This action hierarchy provides a degree of responsiveness in the environment. However, the increased overzealousness may be harmful. Central to the notion of reactivity are the two questions of when and how to react? An intelligent agent ought to react when the utility of reaction is most favorable in light of its goals. The concept of utility is used to construct a measure of desirability for courses of action. An intelligent agent also needs to decide whether to continue with a course of action or whether new circumstances are amenable to a better course of action. To quantify this, each course of action is assigned an expectation of completetion which is monitored continously and updated as new information becomes available. This is tantamount to a feasibility measure of the alternate actions. Choice of motor and sensory activities is determined by arbitration of action at execution while courses of action are discriminated by reasoning at higher cognitive levels.

The Nature of Reactivity

To clarify the different types of reactivity required of a system, we discuss various types of reactivity. We define low-level reactivity as generation of behaviors in response to signals from the environment and identify a need for high-level reactivity.

Reactivity at the low level is a response to *incremental awareness* of environmental variations and details. The prime facie principle is "do the best you can at the moment". Characteristic of low-level reactivity is a goal to be satisfied and a tactic (method) for achieving the goal. Robotic compliant motion is a type of reactivity where fine motion parameters are determined as the environmental constraints are perceived incrementally. An example is sliding along the surface of a table where the geometry of the table is not fully known and are discovered only by sensing local surface geometries. Another type of reactivity is to form responses where the environment including one's resources slowly changes. Activities of the robot can be changed to cope with small changes in the environment. Tracking an object on a conveyor belt is an example.

Low-level reactivity resembles hill-climbing algorithms and is often not sufficient to guarantee successful achievement of goals. Pure low-level reactivity may lead an agent to repetitive actions, undo actions, irreversible traps, or other undesirable situations. We suggest invocation and/or formation of goals with varying strength in response to environmental signals. Most urgent goals might be expanded to methods for accomplishing goals. These methods in turn will be presented to the low-level reactive module for execution. This method of changing goal priorities is a reactive scheme which will be dubbed high-level reactivity.

If one examines the behavior of reactive systems such as human beings, one finds that they react to crisis situations, to situations that impact preserving their good condition, property, and products of their effect, to situations conditined by their profession or role, and to situations that satisfy theur basic needs. These are all examples of high-level reactivity. Purposive robots in a dynamic environment will encounter similar situations and need mechanisms for achieving similar behavior.

BEST COPY AVAILABLE

2. Related Work

A number of low-level reactive systems has been developed in the last few years. Brooks [1985] describes an architecture for incorporating control mechanisms of a robot in specialized behavior units at hierarchies of increasing competence. His recent work has been directed toward building increasingly smaller agents with evolving patterns of behavior. This approach is not appropriate for space applications bacause it assumes resource rich environments rather than resource limited. Vachtsevanos and Hexmoor [1986] present a reactive approach to obstacle avoidance based on rapid replanning capabilities. Agre and Chapman [1987] advocate a model of interaction with the environment that is based on local schemas and demonstrate achievement of complex behaviors without the use of traditional planning techniques using world models. Kaebling [1987, 1988] presents specification languages REX and GAPPS that capture behaviors of agents for parallel actions and provide definition of constructs with constant bound. Agent behaviors are defined in various levels of sophisticaction which provides a hierarchy of behavioral choice based on the scope of available information. Dean and Boddy [1988] present an analysis of time-dependent planning. They introduce a class of algorithms they call "anytime" algorithms which can be suspended and resumed with negligible overhead and which will return an answer whenever terminated. The answers returned improve as a function of time in a well-behaved manner.

A number of planning issues that arise pertaining to reactive systems. For example, reactive systems can be inefficient planners. Drummond [1988] discusses how overzealous reactive systems fail. He presents a problem of stacking three blocks where the middle block cannot be put in place last. He calls this the "B Not Last" (BNL) problem. Plans are expanded in plan nets and situated control rules are provided to avoid potential traps. Monitoring and sensing the environment is also problematic. Gini [1985] describe a system that generates expectations of plans by discovering intents of plan steps. This is used to monitor execution of plans and replan when errors are fully identified. Her recent work [Gini, 1988] clearly points out that "real-time" perception is unattainable and planning based on this assumption is problematic. Planners need to keep track of reasons for plans and their formation in order to reason about and revise them. Very few teleologically adequate planning systems have been developed. An approach toward this end is consideration of mental attitudes like intention, desire, and belief. Georgeff [1987] describes such a need for reactive systems. His work served to demonstrate that intelligent goal-directed behavior in dynamic environments necessitates basing behavior generation on higher cognitive function such as intention. Our recent work in [Underwood and Hexmoor, 1989] is another step in this direction of establishing teleological foundations for planning. Our earlier work uses a schema-based hierarchical planning model. This model has been used in automatic planning of robotic fabrication and assembly tasks in airframe manufacturing [Underwood, et al, 1984 and 1988]. This approach allows modification of prior planning constructs to generate new plans.

3. A Reactive Robotic System Architecture

We propose an architecture for low-level and high-level reactivity that is illustrated in Figure 3.1. The low-level reactive apparatus is similar to previous reactive systems with essential components of perception and an action arbiter. A novelty of our architecture is incorporation of high-level reactivity and a supervisory level of planning and reasoning which guides choice of low-level reactive behaviors.

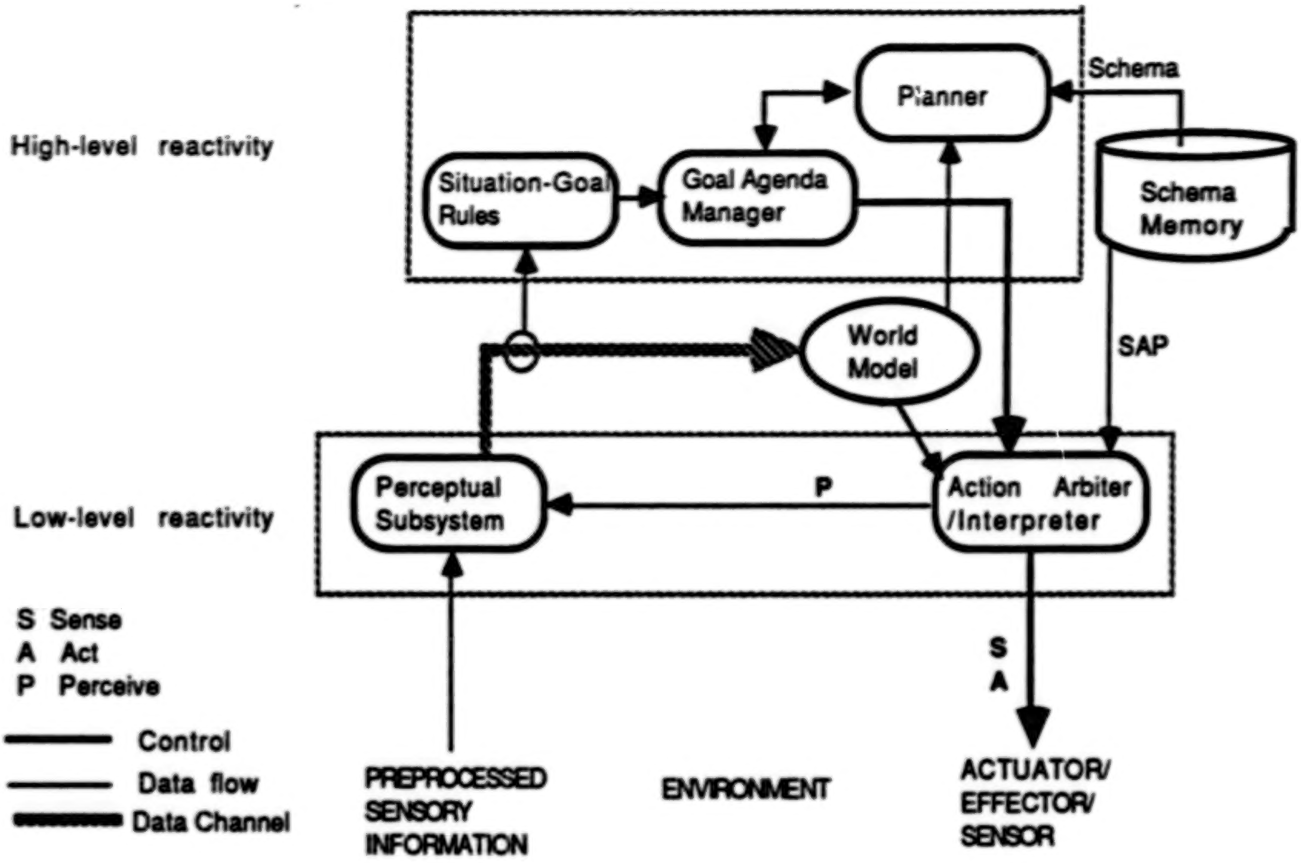

High-level reactivity
Low-level reactivity
Planner
Schema
Schema Memory
Situation-Goal Rules
Goal Agenda Manager
World Model
SAP
Perceptual Subsystem
P
Action Arbiter /Interpreter
S
A
S Sense
A Act
P Perceive
Control
Data flow
Data Channel
PREPROCESSED SENSORY INFORMATION
ENVIRONMENT
ACTUATOR/ EFFECTOR/ SENSOR

Figure 3.1 An Architecture for an Intelligent Reactive Robotic System

The world model and the schema memory are used for both low-level and high-level reactivity.

World model: This component is the database of the perceived environment and contains current information about location of parts and situation parameters along with time tags. Situations are robot activities like grasp and parameters are information used to complete the activities. The world model also contains information about the geometries of solid objects. To preserve internal consistency, only the perception subsystem adds information into the world model. The world model is consulted by the action arbiter and the planner. The world model is also intended to contain information about dynamic environments.

High-Level Reactivity

In this section we describe the objects, structure and mechanisms of high-level reactivity depicted in the upper portion of Figure 3.1.

Goals must be considered in a reactive system because they arise as a reaction to a situation, as a result of planning to achieve a goal. Schank and Abelson [1977] suggest a taxonomy of goals that are useful in automated language understanding. We previously applied this taxonomy in robotic planning and now find it useful is representing high-level robotic reactivity to situations. Particular types of goals of interest are as follows:

o Achievement goals arise from an agents role or function and have to do with the achievement of goals associated with that function.

o Preservation goals have to do with preserving or maintaining the good condition of a system, its possessions, or the results of its achievement goals. For example, P-Condition represents the goal of preserving optimal operating condition.

o Crisis goals are a special case of preservation goals that arise in response to crisis situations, e.g. fire.

o Satisfaction goals are goals corresponding to a recurring strong operational need which when satisfied are extinguished for a time, e.g. , the need for storage of electrical energy.

o Instrumental goals are any goal which facilitate realizing other goals. For example, I-Prep(part, op) represents the goal of preparing a part for an operation.

o Delta goals are distinguished types of instrumental goals that have to do with a change of state. For example, D-Prox(part, loc) represents the goal of changing the proximity of a part to location.

Situation-Goal Rules: Situation-goal rules are used to represent the goals that should be achieved in a particular circumstance. We will discuss various classes of situation-goal rules distinguished by the class of reaction they generate.

Self-Repair or Maintenance Reactions
An example of a situation goal rule in this class is: If malfunction(x), then P-Condition(x). The interpretation of this rule is that if a malfunction is perceived, then pursue the goal of preserving the optimal operating condition of x, P-Condition(x).

Threat Avoidance Reactions
An example of a rule in this class is: If some natural proceess (or other agent's actions) might cause a negative change in operating conditions, then consider the goal of blocking the natural process (or goals of the threating agent).

Another rule in this class is that for preserving possessions or other objects of value. For instance, if an agent has a tool that is useful for achieving its goals, or it has expended time and energy in accomplishing an assembly task, and it perceives a situation that threatens its possession or achievment, then it should consider the goal of preserving possessions or preserving the product of its efforts.

Reactions Conditioned by the Role of an Agent
These rules capture the robotic agent's reactions to situations in which it should respond to achieve some goal for which it was designed. These situations include requests from other agents.

Reactions based on Operational Requirements
And example of a rule in this class is: If Battery-Charge(agent) = low, then S-Energy(agent). This rule represents a reaction to the situation that an agent's battery charge is low. The agent should pursue the goal of satisfying this need for electrical energy.

Goal Agenda Manager: High-level reactivity is initiated by the triggering of situation-goal rules that are monitoring perceived situations that are being passed to the world model from the perceptual subsystem. When triggered these rules pass their associated goal to a goal agenda manager. The agenda manager determines the precedence among new goals and goals currently on the agenda. The precedence of goals is: crisis goals have precedence over satisfaction goals, satisfaction goals have precedence over achievement goals, and achievement goals have precedence over preservation goals. Since instrumental and delta goals arise in planning for accomplishment of any of the goal types above, they inherit the precedence of their ancestor goal.

The agenda manager requests that the planner find a plan for achieving the goal at the head of the agenda. The planner returns an instantiated schema as a sequence of goals, a planbox, or a script. The agenda manager puts this instantiated schema at the head of the agenda and links it to the originating goal for this plan. If the goal at the head of the list is not an instantiated script or planbox, it is still a general subgoal, so the agenda manager will send this goal back to the planner for refinement. Instantiated scripts and planboxes are called methods. When the goal at the head of the agenda is an instantiated script or planbox, the agenda manager will send a goal/method pair, consisting of the instantiated script or planbox and its ancestor goal, to the action arbiter. The action arbiter will be discussed in the section on low-level reactivity. However, in this context, the action arbiter returns a message as to success for failure in achieving the goal. The action arbiter will continue to attempt to achieve success until given another goal. The agenda manager must determine when repeated goal failure amounts to goal blockage. In the case it decides that a goal is blocked, it creates a goal to remove the blocked goal, puts it at the head of the agenda, and requests the planner to consider this new goal.

If the situation-goal rules generate a new goal and this new goal takes precedence over a currently pursued goal, the pursued goal is temporily suspended. A problem that can occur during suspension is that completed subplans and preconditons can come undone. When the agenda manager reactivates a previously suspended subgoal, there is a need to check the current situation against prior achievments to restablish the plan structure.

The goal agenda manager also associates a priority with the goal/method pairs sent to the action arbiter. This is needed to interrupt the operations under the control of the action arbiter when a reaction is required to a goal of high precedence, for example, a crisis goal.

Schema Memory: Schema memory is a data base of all common sense schemas for use by the planner in composing networks of goals and methods necessary for determining appropriate cources of action. Schemas contained in this knowledge base are used to compose a hierarchical plan. Plans constructed from this knowledge are a specification of what to do in the perceived situation. This is unlike 'expectation-based' planners like STRIPS.

There are three types of schemas: scripts, named plans, and planboxes. A script is a plan that has become routine. Named plans are general plans that have worked previously. Planboxes are a more general type of script, intermediate between named plans and scripts.

Structure of schema memory is depicted in Figure 3.2. A-Goals and P-goals are associated with named plans, C-Goals are associated with scripts, S-Goals are associated with named plans and scripts. Named plans are in turn associated with other D-goals and I-Goals, and Scripts are associated with Sense-Act-Perceive (SAP) microcommands. SAPs are discussed in the section on low-level reactivity. D-Goals are associated with planboxes which in turn are associatedwith SAPs. The leaves of the schema structure are all SAPs.

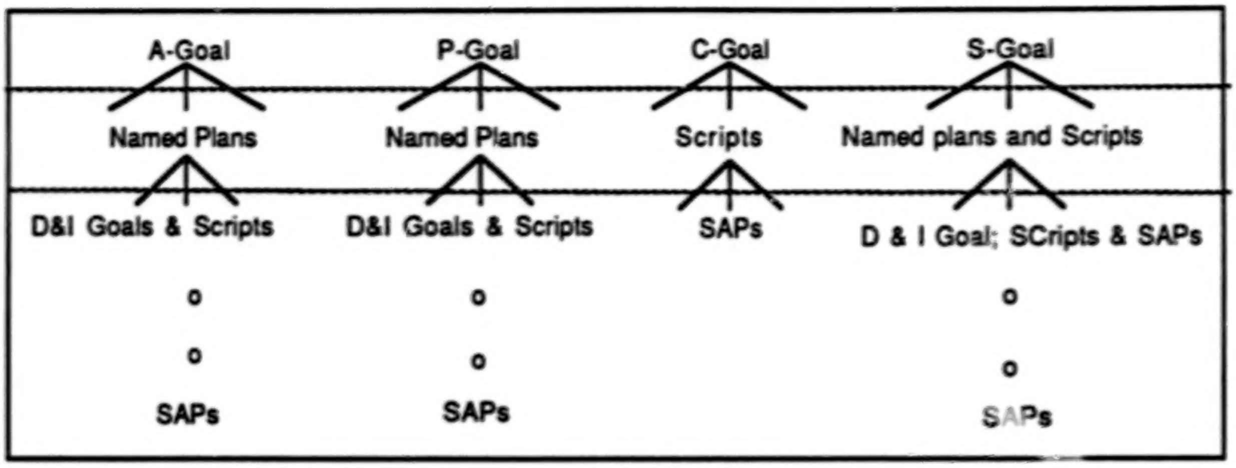


Figure 3.2 Structure of Schema Memory

Planner: The planner responds to requests from the agenda manager to find a plan for achieving a goal by searching the schema memory for relevant named plans, planboxes or scripts. It uses the world model to instantiate these schema and returns them to the agenda manager. The planner responds to a plan request with a single instantiated schema. If the first element of this schema is itself a goal, the agenda manager will request a plan for achieving that goal. Thus the plan is expanded on the agenda. At any point that the schema at the head of the agenda is an instantiated script or planbox, the agenda manager passes that method and its ancestor goal to the action arbiter, rather than requesting planning for goals further down the agenda. Thus the planning will be reactive to situations encountered during execution of the partial plans.

If the planner receives a request from the agenda manager to remove a blocked goal, it will search for an alternative named plan or planbox or it might respecify the goal by substituting a different value for a parameter of the goal. For example, if an instrumental goal is blocked, then the planner might either select a different planbox or substitute a different instrument for this goal. This captures a typical reaction of an agent who is blocked in achieving its goals. When these alternative schema for achieving a goal, a a measure of desirability based on performance can be assigned for selecting among alternatives.

Figure 3.3 illustrates a plan for assembly of two parts as it was expanded on the agenda.

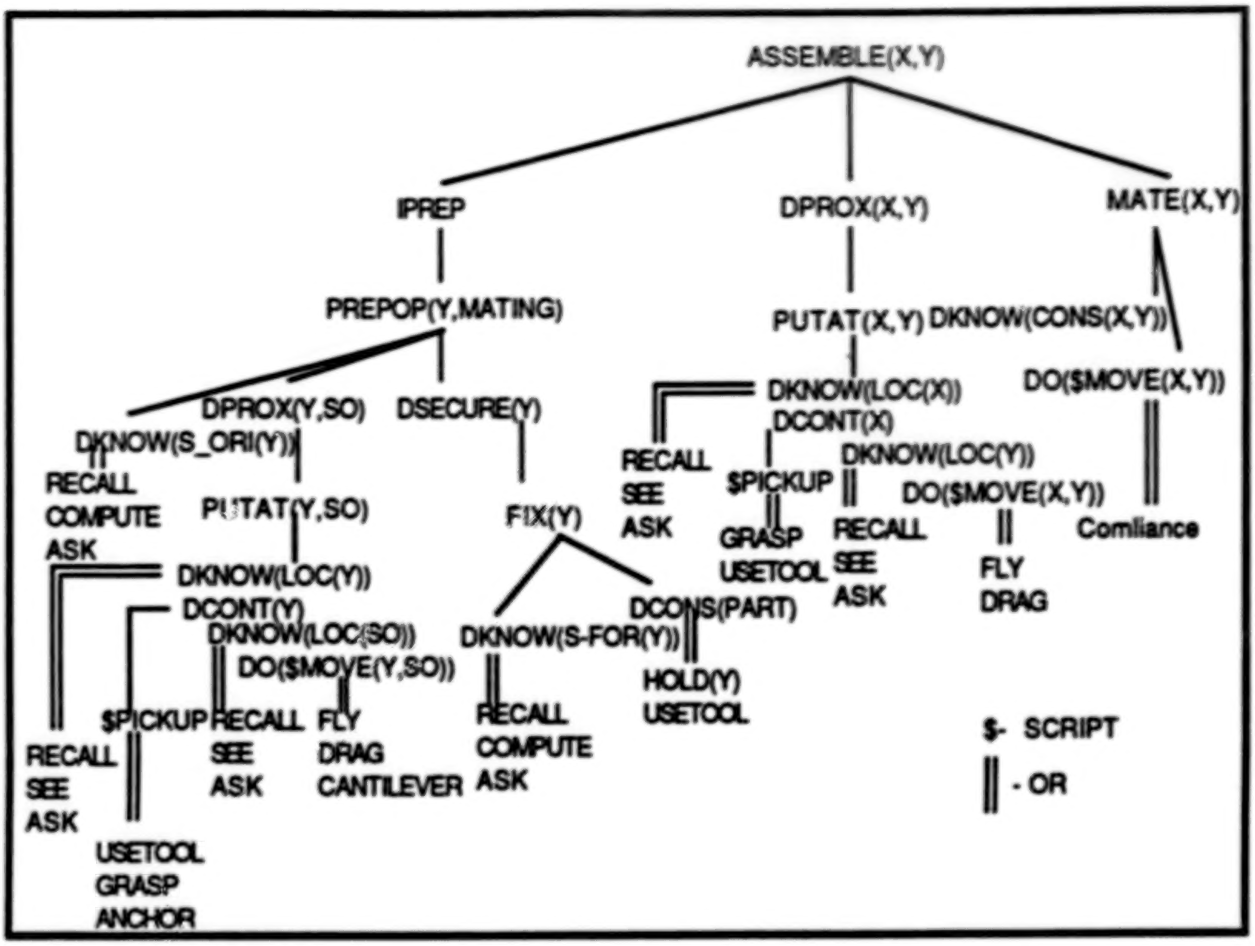

ASSEMBLE(X,Y)
IPREP
DPROX(X,Y)
MATE(X,Y)
PREPOP(Y,MATING)
PUTAT(X,Y)
DKNOW(CONS(X,Y))
DPROX(Y,SO)
DSECURE(Y)
DKNOW(LOC(X))
DCONT(X)
DO($MOVE(X,Y))
DKNOW(S_ORI(Y))
RECALL
COMPUTE
ASK
PUTAT(Y,SO)
FIX(Y)
RECALL
SEE
ASK
$PICKUP
DKNOW(LOC(Y))
DO($MOVE(X,Y))
GRASP
USETOOL
RECALL
SEE
ASK
FLY
DRAG
Comliance
DKNOW(LOC(Y))
DCONT(Y)
DKNOW(LOC(SO))
DO($MOVE(Y,SO))
DKNOW(S-FOR(Y))
DCONS(PART)
HOLD(Y)
USETOOL
$PICKUP
RECALL
FLY
RECALL
SEE
ASK
SEE
ASK
DRAG
CANTILEVER
RECALL
COMPUTE
ASK
USETOOL
GRASP
ANCHOR
$- SCRIPT
‖ - OR

Figure 3.3 Plan for achieving the goal Assemble (X,Y)

Low-Level Reactivity

In this section we describe the structure and mechanisms for low-level reactive behaviors depicted in the lower portion of Figure 3.1.

Perception: This component is responsible for sensory perception of the current state of the environment, including the state of the robotic agent. It operates independently and continuously updates locations of parts and other information. This module must allocate time for processing all sensory input. When it receives a request for finding the location of a particular part, it time shares its computational resources to assess the requested location while continuing to process other sensory information.

Action Arbiter: This component is responsible for generating sensory and motor actions for reactive behavior. It is provided with a goal to achieve and a method for achieving it. Methods are instantiated schemas for achieving goals. It may be given a D-PROX goal and a Pickup script or a PUTAT planbox as a method for achieving D-PROX. Furthermore, methods have distinct phases of operation. Figure 3.4 depicts the structure of a method. An example of a method is a Pickup script with the following four phases: open, approach, grasp, lift.

```
Goal: <goal>
Target: <{specification of rela>
History: {executed-succeeded, executed-failed}
Method: Do
            Phase 1: <Phase 1 SAP target specification>
            Phase 2: <Phase 2 SAP target specification>
            ......
            Phase n: <Phase n SAP target specification>
        End
```

Figure 3.4 Structure of a Method

Sensory, perceptual, and motor actions are packaged into sense-act-perceive (SAP) microcommands and issued in a control loop with their parameters changing continually. Figure 3.5 shows the structure of a typical SAP. Motor commands to the robot are one of the set {Move, Open, Close, Pause, and Halt} each with several parameters. Sensory commands are one of the set {Picture, Imprint, and Force/Torque}. High-level perceptual commands are one of the set {Locate-Object, What/How-Moving}. Targets in a SAP define symbolic relations that need to hold among objects. These symbolic definitions are constructed using a functional assembly specification language we are developing [Hexmoor and Underwood, 1989]. Simple specifications are descriptions of unary or binary conditions/relations among object(s). An example of a unary SAP target is the "condition:open-wide, object:gripper", represented as openwide-gripper. A binary target is "object1:grippper condition:rightside-of object2:book", represented as gripper-rightside-of-book. Parameters for commands are computed by interpolating intermediate increments based on the current values of target specifications in the world model. Static information about object geometry may be found in the world model. These intermediate increments correspond to the resolution of sensory perception. The threshold of this resolution is a prime factor for successful interaction in our environments, natual or man made.

Each SAP contains attributes which help in determining its appropriateness to the current situation. The action arbiter chooses a set of SAPs which have similar goals and methods to the current goal/method and places them in a queue, called the SAP Exeution Queue (SAPEQ). These SAPs are independent of one another. This queue is continuosly monitored for information and resource requirements. All SAPs whose resource requirements match the currently available resources become candidates for execution and are placed in the order of their sophistication on another queue called the SAP Candidate Queue (SAPCQ). The SAP with the highest rating of sophistication is deployed for excution.

```
Method: {scripts, planboxes}
Goal: <Goal-type>
Sensory Resources: {vision, tactile,force/torque,...}
Other Resources: {tools,fixtures,parts,...}
Target: <{specification of relations}>
Sophistication: <Level>
History: {executed-succeeded, executed-failed}
Repeat
    -Using world model update parameters, abnormal termination and suucessful termination
       conditions for current sensory and motor actions for the next increment;
    -Concurrently transmit sensing commands to sensors and
                        transmit motor action commands to robot with priority and
                        transmit perceptual commands to perception module with priority;
    -Compute status of execution
Until (Status = abnormally terminated or successfully terminated)
```

Figure 3.5 Structure of a SAP

Two varieties of low-level reactivity can be differentiated based on the rate of change of the situation. First, the situation might be static or unchanged. SAP parameters are updated incrementally as the environmental contraints are perceived. Compliant motion is an example of this type of reactivity. Failure of SAPs to achieve the goals of their actions in these situations are due to limits of resolution of sensory perception and the effectors.

Senondly, the situation might change slowly enough that the situations are well within direct sensory perception thresholds, so that adjustments of parameters and termination conditions can be computed. Reactivity in these situations amounts to goal refinement. Each SAP is given the capability of incrementally adjusting the actions of all actuators, sensors and the perceptual module. An example of this variety of reactivity is tracking an object on a moving conveyor belt. Some refer to this variety of low-level reactivity as adaptivity.

The situation might change so rapidly that the situations are below the resolution of sensory perception. Adjustments to parameters in these situations might not be adequate to achieve the goal, so that new goals might have to be substituted for old ones. In these cases new SAPs will be selected.

Upon termination of a SAP, status of the SAP is examined in light of the overriding goal, the method and the changes in the environment. This status is one of the set {terminated normally, terminated abnormally}. In the special case of a static world, a retrial of the same SAP is issued, and in case of repeated failure, supervisory levels of control will intervene. Otherwise, depending on how much the world has changed, SAP queues are reprocessed. Figure 3.6 summarizes these relations in a decision matrix.

	SAP succeeded	SAP failed
Situation Unchanged	Pick the next Goal/Method*	React1: try again
Situation Changed Slowly	Pick the next Goal/Method **	React2: Update SAPCQ and retry
Situation Chaged Rapidly	Pick the next Goal/Method ***	React2: Update SAPCQ and retry

* No reaction was necessary
** Reacted by changing SAP parameters in-flight
*** Probably unrelated changes
React1:First variety of reaction
React2:Second varity of reaction

Figure 3.6 Action Arbiter Decision Matrix

Figure 3.7 summarizes the activities of the action arbiter. Note that with the availability of most information and resources, execution proceeds with the most certain scheme of accomplishing the task transmitted to the robot. We refer to this as executing at level 1. Otherwise, conditions are tested for a less certain version of accomplishing the same task.

```
Repeat
      -Select the next goal/method
      -From schemas select SAPs with similar goals and methods to the current goal/method and
       put them on the execution queue (SAPEQ)
    Repeat
            -From the execution queue select SAPs requiring resources similar to currently available
             resources and put them on SAPCQ in the order of their sophistication
       -Execute the SAP with highest sophistication from the SAPCQ which
           may not have failed more than once
          -Remove SAP from SAPEQ if SAP failed twice
     Until (Empty SAPCQ or Empty SAPEQ or SAP is terminated successfully)
    -Empty SAPEQ
     -Generate status report and send to agenda manager
Until (no goal/method)
```

Figure 3.7 Action Arbiter

The action arbiter may receive interrupts from higher levels with three levels of priority. The lowest priority interrupt will cause a pause in the motor actions. The next higher priority interrupt may direct the action arbiter to a different goal/method queue. The highest priority interrupts may be sent through the arbiter to the robot to stop an overzealous low-level reactivity. To handle high priority interrupts a service routine is often necessary. For example, a high priority interrupt might activate a service routine to grip an object more tightly to prevent it from slipping from the gripper while the object is being moved. Service routines are types of SAPs and there is a supply of service routines which are scheduled for execution and are transmitted to the robot with the highest priority to redirect the activities of all actuators and effectors as well as sensors and the perceptual module.

4. Implementation Issues

We are using a six degree of freedom Universal Machine Intelligence RTX robot in prototyping this conceptual design. We are adding sensory capabilities of force/torque sensing at the wrist, a vision system with a camera looking down on the workspace, and a tactile sensor at the inside of the gripper jaws.

The task of spatial reasoning in the perceptual module is by far the most difficult to implement for real-time reactivity. Vision and tactile sensory preprocessors generate grids of data for intepretation. Each image needs to be understood individually and correlated with other sensory data for recognition of situations. One approach to implementation is to use a hierarchical perceptual processing system, much like a blackboard model of problem solving, where specialized processing modules intepret data and form perceptual hypotheses for higher perceptual modules. At the topmost level patterns of perception trigger percept schemas that update the world model along with a time stamp. Since a reactive system requires efficient perception, one might opt for additional processing power, but in resource poor space applications this might not be a feasible alternative. Directed perception and improvements in perceptual algorithms are a more likely implementation alternatives in resource poor applications.

Implementation of high-level reactivity requires a flexible control framework. This might also be achieved through a blackboard architecture with a control blackboard [Hayes-Roth 1988, Underwood, et al 1988]. However, more efficient implementations of this framework will be necessary to support the real-time requirements of reactivity.

A SAP can be considered as a feedback loop where commands are generated based on the previous state of the world. Figure 4.1 show a SAP's feedback loop. In each cycle perception commands are sent to the perceptual module, sensory commands to the sensors, and motor action commands to the robot controller. The world model is updated and new parameters are computed. As argued by Gini [1988], it is not realistic to assume rapid perception. With the advent of smart sensors, it is possible to delegate elementary monitoring tasks to the sensors. This enables inclusion of an embedded feedback loop within the sensor with faster sampling rates in the order of milliseconds versus seconds for the outer SAP feedback loop.

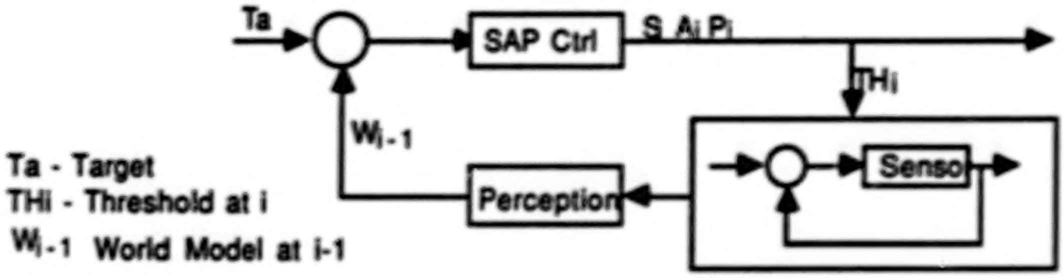


Figure 4.1 SAP as a Feedback Loop

The sampling rate of SAPs are adjusted for types of motion. Gross motions need lower sampling rates, versus fine motions where more rapid samplng os required.

5. Summary and Conclusion

Autonomous systems must react and adapt to situations in their environment. Two levels of reactivity should be distringuished. High-level reactivity to situations is at the conceptual level. Low-level reactivity is at the sensory, perceptual and motor level. Whereas other considerations of reactive systems have addressed the latter type of reactivity, we propose an architecture that can realize both levels.

High-level reactivity can be realized by a continuously active perceptual system, situation-goal rules that are triggered by perceived situations, an agenda manager that schedules consideration of new goals in the context of current and pending goals, a schema-based planner that suggests appropriate patterns of behaviors for achieving goals, and by capabilities for low-level reactivity.

Low-level reactivity adopts the principle of "do the best you can at the moment" and is tantamount to either adapting a motor action or substituting motor actions at different levels of competence to accomplish a given tactical goal. Motor acts are adapted to a situation by specifying symbolic targets for motor acts and adjusting their parameters including their termination conditions. In a changing environment, methods for accomplishing tasks can change in appropriateness and the availability of different means of accomplishing the same task affords a higher degree of reactivity.

A significant reaserch direction is to integrate reactive planning with learning capabilities that are cognizant of effects of action over time. These systems might learn patterns of failure and success and generalize plans. An extension to reactive systems is to consider multi-agent reactive systems and have them reason about other agents instantaneous behaviors.

Automation of assembly and repair tasks is difficult and abundant uncertainties indicate flexibility in adjusting to the environment is necessary. Robotic assembly in space will require robust systems that can react to situations. Although much of the discussion in this paper has addressed the domain of sensor-based robotic assembly, the architecture and techniques developed are applicable to mobile robots, such as the mars rover, and many other space systems that could benefit from a higher degree of autonomy .

References

Agre, P.E. and Chapman, D. (1987), "Pengi: An Implementation of a Theory of Activity", In *Proceedings of AAAI 87*, pp. 286-272.

Brooks, R. (1985), "A Robust Layered Control System for a Mobile Robot", A.I. Memo 864, MIT AI laboratory.

Dean, T. and Boddy, M. (1988), "An Analysis of Time Dependent Planning", In *Proceedings of AAAI 88*, pp. 49-54.

Drummond, M. (1988), "Situated Control Rules", In the *Proceedings of the Rochester Planning Workshop*.

Georgeff, M., Lansky, A.L., and Schoppers, M.J. (1987), Reasoning and p[lanning in Dymamic Domains: An Experiment with a Mobile Robot", TN 380, SRI International.

Gini, at al, "Symbolic Reasoning as a Basis for Automatic Error Recovery in Robots", TR 85-24, Computer Science Department, University of Minnesota.

Gini, M. "Automatic Error Detection and Recovery", TR 88-48, Computer Science Department, University of Minnesota.

Hayes-Roth, B. (1988), "Making Intelligent Systems Adaptive", STAN-CS-88-1226, Department of Computer Science, Stanford University.

Hexmoor, H. and Underwood, W. (1989), "An Ontology and Repersentation for Flexible Assembly", In preparation.

Kaebling, L.P. (1987), "An Architecture for Intelligent Reactive Systems", In Michael P. Georgeff and Amy L. Lansky, editors, **Reasoning about Actions and Plans**, pp. 395-410, Morgan Kaufman Pubishers.

__, (1988), "Goals as Parallel Specifications", In *Procedings of AAAI 88*, pp. 60-65.

Kim, S. (1988) "Information Framework for Robot Design", In International Encylopedia of Robotics, Richard C. Dorf, editor, Wiley and Son Publishers, NY.

Liu, Yanxi andArbib, M.A. (1986), "A Robot Planner in the Assembly Domain", COINS TR 86-36, University of Mass., Amherst, MA.

Schank, R. and Abelson, R. (1977), "Scripts, Plans, Goals and Understanding", Lawrence Erlbaum Associates , Inc., Publishers, Hillsdale, NJ.

Underwood, W. and Hexmoor, H. (1989), "Intentional Conepts in Industrial Management and Engineering", Submitted to The Second International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Tullahoma, Tennessee.

Underwood, W., MacGreggor, B.K., and Scruggs, R.M.(1984), "Scripts, Plans, Goals and Robot Planning", *Focus on Manufacturing Technology Research*, Georgia Institute of Technology, March 1984.

Underwood, W., Hexmoor, H., and Scruggs, R.M. (1988), "Intelligent Task Planning and Execution for Robotic Control", Technical Report AIAI-88/01, Artificial Intelligence Atlanta, Inc., Decatur, Georgia.

Vachtevanos, G., Hexmoor, H., "A Fuzzy Logic Approach to Robotic Path Planning with Obstacle Avoidance", *Proceedings of the International Conference on Control, Decision*, Athens, Greece, 1986.

PST & PARR: Plan Specification Tools And A Planning And Resource Reasoning Shell For Use In Satellite Mission Planning

David McLean and Wen Yen

Software and Engineering Services
Bendix Field Engineering Corporation
Seabrook, Maryland

ABSTRACT

PST (Plan Specification Tools) is a set of tools which allows the user to specify satellite mission plans in terms of satellite activities, relevent orbital events and targets for observation. The output of these tools is a set of knowledge bases and environmental events which can then be used by PARR (Planning And Resource Reasoning shell) to build a schedule. PARR is a reactive planning shell which is capable of reasoning about actions in the satellite mission planning domain. This paper describes each of the PST tools and PARR and then briefly describes the use of PARR for scheduling computer usage in the multisatellite operations control center at Goddard Space Flight Center.

INTRODUCTION

PST has evolved through need in response to the problem of building planning systems which can be built quickly and which can respond to changing requirements after the initial system has been delivered. PST generates planning knowledge bases and resource viewing periods which can then be used by PARR. Experience with the ERBS-TDRSS (Earth Radiation Budget Satellite - Tracking and Data Relay Satellite System) Contact Planning System [McLean, 1987] has revealed the need for easy to use tools which allow operations personnal versus knowledge engineers to make changes to the planning knowledge bases when unforseen (testing) situations arise. In the future, some satellite users (Extreme Ultra Violet Explorer) will make a preliminary survey and then make ad hoc selection of targets to be observed. PST is a step toward satisfying both of these requirements. In its present form, PST consists of an event specification tool, a tool to predict the satellite orbit positions, a target selection tool and a tool to generate viewing periods for resources and targets.

PARR is a single tool which can be invoked in a batch or interactive mode to generate a schedule for the events specified by PST. PARR is a reactive [Ow, 1988] planning shell because it has the ability to react intelligently to conflicts as they are encountered. This reactive component of PARR and the fact that the output of PARR is an actual conflict-free schedule makes it a tactical planning tool. On the other hand, PST can be thought of as a set of strategic tools because PARR is used to execute the "strategic plans" (planning knowledge bases and environment) generated by the PST tools. These tools are integrated by a Menu-based EXecutive (MEX) which is based on the notion of menu-based natural language understanding introduced by Tennant [1983]. Figure 1 shows the organization of these tools.

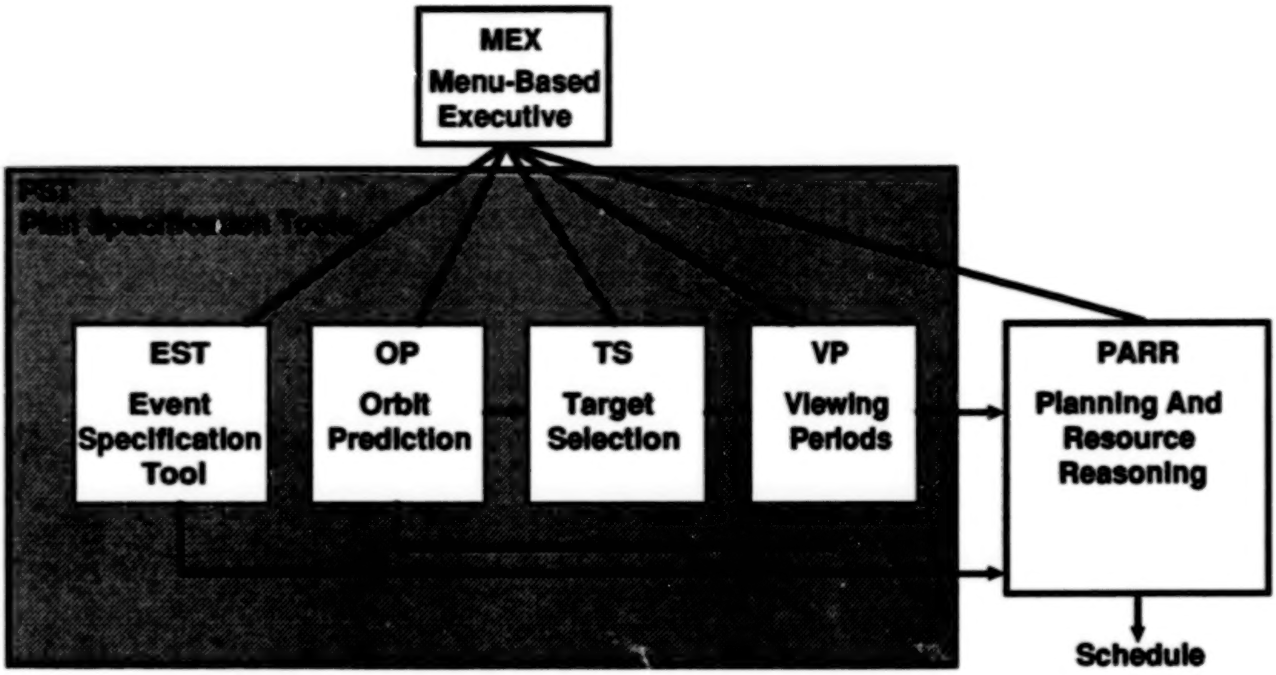

MEX
Menu-Based
Executive
PST
Plan Specification Tools
EST
Event
Specification
Tool
OP
Orbit
Prediction
TS
Target
Selection
VP
Viewing
Periods
PARR
Planning And
Resource
Reasoning
Schedule

Figure 1 Planning Tools Environment

All of these tools are written in C and therefore are easily portable to a large variety of conventional hardware. Each of these tools will now be described and then an example using PARR for scheduling satellite equipment will be given.

THE EVENT SPECIFICATION TOOL

Both the MEX and the Event Specification Tool (EST) use a library of menu tools which allow the system designer to specify a grammar of valid commands. When using the menu tools, the designer specifies a top level menu which consists of one or more options (lines of text) each of which consists of a number of tokens (words delimited by spaces). Each of these tokens may then be the name of another menu. As options are selected, tokens which match the name of other menus get expanded into further menus and those which do not get added to the command line being built. This simple paradigm, combined with a few basic menu types, results in a very powerful tool for command line building. Some of the nonstandard menu types which are typically used by EST are a matrix type which allows the designer to arrange the menu options in a matrix (two dimensional array) and a get-typed-text type which accepts text typed from the keyboard. A complete description of how to use the library of menu types is described in the MEX user's guide [1988].

The top level menu used by EST specifies the general plan specification model to be used. Currently only one model is used to specify events and this model includes the following components:

1. The spacecraft event name.
2. How often the event is to be scheduled (every N orbits).
3. The duration of the event.
4. When to start the event.
   (usually with respect to a specified resource window)
5. Alternative means of scheduling (if any).
6. Spacecraft resources (tape and power, if any).
7. Constraints (if any).
8. Miscellaneous parameters such as priority and offset.

The actual top level menu looks like this:

```
main
   s/c_event every_n duration start_event any_alternatives \
   any_resources any_constraints misc_parms
```

Note that there is exactly one token for each component of the model (the backslash character indicates line continuation). Each of these tokens is a menu name which gets expanded when EST is invoked, for example, the following menus get used when the "start_event" token is expanded:

```
start_event
     title: Select the start time type you wish to specify.

     at_orbit_event_point
     user_specified_start
.
at_orbit_event_point
     title: Select START or END of the orbit point.

     START orbit_event
     END orbit_event
.
orbit_event
     title: Select an orbital view period.

     Daylight
     SAA
     MA_TDRSS_viewing_periods
     SSA_TDRSS_viewing_periods
     .
     user_specified_start
          prompt: Enter a start time (HH:MM:SS):
     .
```

The "title:" token is used to specify text which is to be used to label the menu. The "prompt:" token is used to specify text to prompt the user for typed input. The actual number of user-specific menus, such as "s/c_event" and "orbit_event", required for an application using this model is small because most of the menus are generic to the model without being user specific.

The knowledge base structure which results from the use of EST is an event frame with slots for each component of the model. These components have been described elsewhere [The IEPS Knowledge Base Author's Manual, 1986], but an example of such an event frame demonstrates the correspondence between it and the current EST model:

```
     TDRS_W_Tape_Dump_via_MA_antenna
     ;    priority
               3
     ;    repeat every N views
               2
     ;    duration
               0:30
     ;    offset from view start
               0:00
     ;    allow shift (if subsequent conflict occurs)
               YES
     ;    resources:
               LCS_POWER 200 1000
               CR_TAPE -1 1500
          .
     ;    constraints:
               inview eq TDRS_W_MA_viewing_periods
               inview ne South_Atlantic_Anomoly
               activity eq NONE
          .
     ;    event start:
               START TDRS_W_MA_viewing_periods
     ;    alternative strategies:
               EVENT TDRS_W_Tape_Dump_via_MA_antenna
               NEXT 1
          .
     ;    substructure
               NONE
     ;    display text: (text displayed to user on expansion)

               This is a tape dump using the
               TDRS west multiple access antenna.
```

More detailed explanation of some of these slots will be described in the section on PARR.

THE SATELLITE ORBIT PREDICTION TOOL

The Flight Dynamics Facility (FDF) at Goddard provides accurate positional information for each of the standard orbital events, such as satellite daylight and TDRS viewing periods. However, because future missions will have the requirement of predicting where and when ad hoc targets of opportunity will be in view, some means of predicting viewing periods for these events is necessary. The satellite orbit prediction tool is the first step toward solving this problem. This tool is used to create a file which contains the positions of a satellite in an inertial geocentric coordinate system. These positions are generated at one minute intervals for a period of up to one week. This information is then used by the target selection tool and the resource and target viewing period tool to generate the necessary viewing periods for a variety of orbital phenomena.

The input to the satellite orbit prediction tool is a recent set of Brouwer mean elements obtained from FDF. The satellite positions are then predicted using Brouwer's first order secular perturbations due to an aspherical earth. For low orbiting satellites, such as ERBS, an empirical term simulating the effects of atmospheric drag is added. The accuracy of these predictions (for ERBS) is within 30 seconds after a week's set of predictions. Greater accuracy can be achieved by obtaining satellite orbit predictions directly from FDF, however a great deal of scheduling can be done with the lower level of accuracy.

THE TARGET SELECTION TOOL

The intent of this tool is to show some general ideas and capability which could be refined and expanded for actual mission requirements. The target selection tool consists of a user catalogue of potential targets (currently only fixed star-like targets are supported) and graphics software to display and select the targets. When the tool is invoked, the user is presented with a screen displaying the entire sky and all the user's targets which are visible on a given date and time. Those targets which are occulted by the earth are not visible. In addition, the position of the sun and the current slew point (instrument viewing direction) is displayed. The user may then step forward in time until the desired time of observation is reached and then select a subset of targets for planned observation by creating a window which contains the desired targets.

Once the targets are selected they are connected by lines which represents an "optimum" slew path. This slew path is arrived at by using a Hopfield net [1986] and various heuristics which result in an acceptable solution to the traveling salesperson problem. At this point, slew times (based on relative right ascention and declination) and exposure times (based on magnitude) are calculated for each selected target. The targets are then written to a file in the optimum slew order with their respective slew and exposure times. This output file of selected targets can then be processed by the resource and target viewing period tool to generate viewing periods.

THE RESOURCE AND TARGET VIEWING PERIOD TOOL

The resource and target viewing period tool uses the information generated by the satellite orbit prediction tool and the target selection tool and generates viewing periods for selected orbital events and targets. These viewing periods become the environment within which PARR must expand the strategic plans (event frames) produced by EST. The South Atlantic Anomonaly (SAA) position and size, orbit number, and the mean elements for TDRS east and TDRS west are obtained from FDF at Goddard. The position of the sun is calculated from formulae given in the Astronomical Almanac [1989]. In addition, the right ascention and declination of the selected targets is available and the user specifies which orbital events are required for scheduling (such as TDRS, Daylight and SAA) by menu selection. From this information, and from the previously calculated satellite positions, the viewing periods for each of these orbital events can be calculated.

The resource types available with this tool include the orbital events listed above and additional spacecraft resources such as power and tape. In the current prototype, spacecraft resources are initally assumed to be completely unused and available throughout the entire duration of the planning interval. A more detailed description of how spacecraft resources are represented is described in the section describing PARR.

The output file created by this tool lists the name of each resources preceeded by the key letter 'R' and each of the targets preceeded by the letter 'T'. These key letters let PARR know what the general resource type is so that when it executes the strategic plan the appropriate actions will be applied. Additional information follows each resource name depending upon the type, for example, slew and exposure times for each targets. On the lines after each event name are listed the date, start and stop times and any additional information such as the orbit number.

THE PLANNING AND RESOURCE REASONING SHELL

PARR is the product of an evolving planning shell which has been in use by ERBS since May of 1987. Its capabilities have therefore evolved from the requirements set by its actual use as well as from the anticipated needs of future users. There are currently two versions of PARR, one which runs on an IBM-PC using character graphics and another which runs on a UNIX-based workstation using bit-mapped graphics via X-Windows. PARR has three main modes of operation: batch, merge and interactive. In the batch and merge modes, the system builds a conflict-free schedule from the strategic knowledge bases alone, without consulting the user. The merge mode differs from the batch mode in that it is used to merge independent previous schedules. In the interactive mode PARR behaves as an intelligent assistant which not only handles the tedious details of planning requests but also suggests expert means for resolving conflicts when they arise. A great deal of the recent capability of PARR is due to its ability to reason about resources. The emphasis of this section will be to describe these recent capabilities.

Reasoning about resources requires that the various resource types be identified according to their behavior and then that a response (action) be selected which is most likely to solve the problem in view of the behavior of the resource. Therefore, a resource taxonomy which defines the possible behavior types is useful. Consider some typical resource examples used in spacecraft scheduling: sunlight availability, power for activities, fuel for manuvering, tape for recording data and targets to observe, such as stars. These resources can be classified as follows:

Resource Example	Resource Type	
Sunlight	Unlimited Capacity Subscribable	(UCS)
Power	Limited Capacity Subscribable	(LCS)
Fuel	Consumable and Nonreplenishable	(CN)
Tape	Consumable and Replenishable	(CR)
Star	Target	

A UCS resource assumes that its presence alone is all that is required for its use. In the early stages of planning, TDRS availability is also a UCS resource because there is no knowledge of other spacecrafts' use of the system. LCS resources can support multiple requests with varying usage rates in parallel up to a threshold level of usage. For example, a tape recorder may use power at the same time that an instrument's sensing device is using it. CN can be considered a special case of the CR type, the case where replenishment is never scheduled. The CR resource type which PARR supports assumes that any number of requests can be supported in parallel and that they all consume at the same rate (as in a multichannel tape recorder). The resource type Target is really a special case of UCS and because targets require special treatment, a separate class has been made. The scheduling of targets is complex and therefore the discussion of this resource type will be differed until after the other types have been described more fully. Additional hybrid resource types can be constructed from combinations of the above; for example, battery power and "coolness" (a thermal sink) are a combination of LCS and CR because they behave like LCS resources which must be replenished from time to time.

Because PARR allows the user to specify resource attributes such as maximum capacity and usage rate, it can use this information as implicit resource constraints. Thus because PARR supports the above resource types, the user need not bother specifying the constraints for each resource explicitly. Normally, PARR uses these implicit constraints (such as, is the maximum capacity exceeded) when activities are requested by a user and rejects requests when these constraints are violated. However, because PARR can reason about these resource types, it can also try actions implicitly, such as replenishment when a CR resource is too low. Further, PARR frees the user from keeping track of the usage rates for each resource but allows the user to see the actual usage levels in the current situation.

The following discussion refers to the slots specified by the event frame and therefore it may be useful for the reader to refer the EST frame example given in a previous section. A UCS resource type is an example of a "viewing period" and because of this, a more traditional approach has been taken to represent this typical planning construct. In the current version of PARR, UCS resources such as sunlight may also be triggering events which are used to specify when an activity is to start. Thus, the name of the specific UCS resource may also appear as part of the "event start" slot. UCS resources are considered external to the spacecraft and therefore, if a particular spacecraft event requires that a UCS resource be available during the entire activity it should also be specified explicitly in the "constraints" slot. Because the constraints slot uses a syntax which specifies attribute-relation-value triplets, it is also useful for specifying which UCS resources should not be in view, such as the South Atlantic Anomoly.

Spacecraft resources which are internal (or on a platform) are specified in the "resources" slot and include the LCS and CR resource types. When specifying these resource types the user must prefix the resource name with "LCS_" or "CR_" so that PARR knows which type is being specified. The first parameter that follows the resource name is the usage rate and the second parameter is the maximum capacity (both are in user-defined units, CR usage is in units/minute). A negative usage rate for a CR type indicates a replenishment activity. Any number of resource types can be specified in this slot but the order is important if the first part (first three characters) of the unprefixed resource names match. This allows the user to specify alternative resource types to be used if the initial types are unavailable. Thus resource names such as the following are typical:

LCS_Power_A
LCS_Power_B
CR_Tape_A
CR_Tape_B

In this example, PARR will implicitly try obtaining the alternative resource "LCS_Power_B" if "LCS_Power_A" is unavailable.

Although the LCS resource type is sophisticated because of its ability to handle overlapping time requests, the CR type has a number of properties which make its implementation even more complicated. First, the amount of resource (for example tape) available must be known before and after each usage and each replenishment. The complication starts when a user wishes to interactively insert new activities which require tape. When this request is made, there may be plenty of tape to support the new request but not also the subsequent requests which have been previously scheduled. This condition (not enough tape for subsequent events) will result in PARR trying the implicit action "replenish". If replenishment is accomplished and all other event constraints are passed, then the request is accepted, otherwise it is denied. Another complication occurs when the user wishes to delete a replenishment. PARR will allow a deletion only if all subsequently scheduled events can be accomplished without the replenishment.

PARR represents targets as having the attributes slew time and exposure time which require special actions. The attribute slew specifies a duration during which an activity called slew must be scheduled before the actual "observation" can occur. The attribute exposure specifies the length of time that the target must be observed. Exposure time may be accumulated by scheduling the observation for multiple orbital viewing periods. PARR also has the ability to schedule multiple targets in a series, including slews and automatic tape replenishment. To specify the "observe targets" action in PARR, the user needs to have specified the slew and replenish (if required) events in addition to the observe event whos primary action will be indicated by the keyword TARGETS in the "event start" slot.

In addition to the implicit actions used by PARR to build the schedule specified in each of the event frames (goals), the user may also explicitly specify a set of heuristics which specify alternative actions (strategies) which may accomplish each goal. This combination of implicit and explicit actions and constraints is what makes PARR so powerful.

MSOCC EQUIPMENT SCHEDULING USING PARR

Part of the task of the Multisatellite Operations Control Center (MSOCC) is to schedule equipment (computers, etc) to support real time contacts for a number of different satellite missions. This effort also includes scheduling the maintenance and offline testing for this equipment. To accomplish this, the MSOCC schedulers must first schedule all the real time support required for each of the missions and then schedule the maintenance and offline testing around the real time events. A further complication is that equipment assignment is changed weekly in order to insure that the same equipment units are not used to support the same missions week after week. This equipment change is necessary so that the reliability among each of the units within a given equipment type is maintained. When a particular unit fails a mission must be shifted to another unit which is equally reliable.

The task of entering the real time requests for support is done on a daily basis and is now accomplished by using a menu-based data entry system (based on the MEX library). Once this is done for each of the missions, PARR is used to assign equipment to each request and then merge all the real time requests into a single conflict free schedule. The equipment assignment is done by creating a knowledge base that represents the pattern of equipment assignment required for a particular week. There are eight possible equipment assignment patterns and therefore eight different knowledge bases. PARR represents the equipment as LCS resource types so that it is possible to support more than one mission on a particular piece of equipment. The user of the system selects the particular weekly pattern required before PARR is invoked and when conflicts of resources occur, PARR uses the alternative resources according to a particular mission's resource list. Any unresolvable conflicts are displayed to the user who may then reenter alternative real time requests. After this automatic assignment is done by PARR, the user may then "fine tune" the schedule by interactively moving resources for any desired mission event.

When the MSOCC schedulers are satisfied with the real time event schedule, a report generator is then invoked which prints each of the realtime events on timelines by equipment type and unit number. This report allows the schedulers to plan the maintenance and offline testing activities as required. After these activities are planned, they can also be put into a database via the same data entry tool as were the real time events. The only difference in data entry is that a different set of menus is used to interact with the user. Once all of the maintenance and offline activities have been entered, PARR is then invoked again to merge these activities with the merged real time events. Again PARR lets the user know if there are any conflicts that it cannot resolve. At this point the user can again make refinements to the merged schedule by interactive use of PARR until the schedule is satisfactory and then regenerate the timeline by equipment reports for further inspection. When schedules have been developed for each day of the week, another report generator is invoked which displays each of the maintenance and offline teams (by name) along with their respective schedules for all the different equipment types they support.

A great deal of the success of the MSOCC equipment scheduling system is due to the flexibility of the data entry and report generation tools. However, it is interesting to note that PARR is general enough to be used in the "equipment scheduling" as well as in the "satellite scheduling" domain.

CONCLUSIONS

The development of these strategic (PST) and tactical (PARR) planning tools has proceeded with a number of goals in mind. Foremost is that these tools should be available and be of use to a variety of different users which have planning and scheduling requirments and which have at their disposal conventional hardware such as IBM-PCs or UNIX-based workstations. In addition, that these tools will evolve in a way that the user interface technology will remain relatively independent of the AI and algorithmic technology so that each can be maintained independently. This allows the greatest flexibility for taking advantage of new technology as it arrives. Finally, that the capabilities demonstrated by these tools should be derived from current mission support as well as the requirements for future support.

Most of the tools described above are not just prototypes but are in actual use. The ERBS-TDRSS Contact Planning System has recently been upgraded to support TDRS west requests as well as TDRS east and to use MEX and PARR. This has led to an enhanced ability to support ERBS because the complexity of specifying how constraint checking and resolution is to be accomplished has been greatly simplified and made more implicit. The fact that these tools are evolving through actual mission support as well as by prototyping features for future support gives credibility to this approach.

ACKNOWLEDGEMENT

The authors whish to thank Patricia Lightfoot and Carolyn Dent at NASA-GSFC/Code 514 and Ellen Stolarik at Bendix Field Engineering Corporation for their continual support of this work. This work was supported by NASA contracts NAS5-31000 and NAS5-27772.

REFERENCES

The Astronomical Almanac for the year 1989, U. S. Government Printing Office, Washington, D. C., 1989.

IEPS Knowledge Base Author's Manual. NASA-GSFC/Code 514, June 1988.

Hopfield, J. J., and Tank, D. W. "Computing with Neural Circuits: A model". Science, vol. 233, August 1986.

McLean, D. R., Littlefield, R. G., and Beyer D. S. "An Expert System for Scheduling Requests for Communications Links Between TDRS and ERBS". Telematics and Informatics, Vol. 4, No. 4, 1986.

MEX: A portable Menu-Based Executive. NASA-GSFC/Code 514, November 1988.

Ow P. S., Smith, S. F., and Thiriez, A. "Reactive Plan Revision". Proceedings of the AAAI-88 Seventh National Conference on Artificial Intelligence, Vol. 1, August 1988.

Tenant, H. R., Ross, K. M., Saenz, R. M., Thompson, C. W., and Miller, J. R. "Menu-based Natural Language Understanding", Proceedings of the Association for Computational Linguistics, MIT, June 1983.

AN APPROACH TO KNOWLEDGE ENGINEERING TO SUPPORT KNOWLEDGE-BASED SIMULATION OF PAYLOAD GROUND PROCESSING AT THE KENNEDY SPACE CENTER

by

Shawn McManus, Michael McDaniel
McDonnell Douglas Space Systems Company
Kennedy Space Center
P. O. Box 21233
Kennedy Space Center, FL 32815

ABSTRACT

Planning for processing payloads at the Kennedy Space Center (KSC) has always been a difficult and time-consuming task. With the advent of Space Station Freedom and its capability to support a myriad of complex payloads, the planning to support this ground processing maze involves thousands of man-hours of often tedious data manipulation. To provide the capability to analyze various processing schedules, McDonnell Douglas Space Systems Company-KSC is developing an object oriented knowledge-based simulation environment called the Advanced Generic Accommodations Planning Environment (AGAPE). Having nearly completed the baseline system, our emphasis in this paper is directed toward rule definition and its relation to model development and simulation. We focus specifically on the methodologies implemented during knowledge acquisition, analysis, and representation within the AGAPE rule structure. An example model is provided to illustrate the concepts presented. Our approach demonstrates a framework for AGAPE rule development to assist expert system development.

I INTRODUCTION

Space station payload ground processing at KSC requires a great deal of planning and analysis, and AGAPE was designed primarily to support this activity. As the system has matured, its capabilities have become quite robust, making the system adaptable to modeling development activity in a wide variety of domains. To highlight and clarify the discussions in the ensuing sections, the major sections will feature a discourse of the relevant topics, followed by two examples from KSC space station ground processing. A future facility at KSC, the Space Station Processing Facility (SSPF) will provide an example of the facilities under consideration in our work, and installing a payload into a rack will benchmark a typical processing sequence.

Some local KSC definitions are in order:

payload - in the context of this paper, this will mean any hardware processed at Kennedy to become part of Space Station Freedom

experiment - any user (i.e. non-system) payload

APAE - Attached Payload Accommodation Equipment, the equipment that allows an attached payload to interface with station resources

II SYSTEM OVERVIEW

AGAPE is an knowledge based, object oriented simulation environment. The system's objects are built in a frame structure, a convenient method to allow storage of object attribute values in structures called *slots*. The object oriented approach to programming defines objects into two categories, *classes* and *instances*. Class objects may have any level of descendents (children, grand-children), while instance objects can have no children. This parent-child relationship allows for *inheritance* of attributes and other structures to be discussed later. This hierarchy constructs a tree-like *genealogy* of objects, as seen in Figure 1.

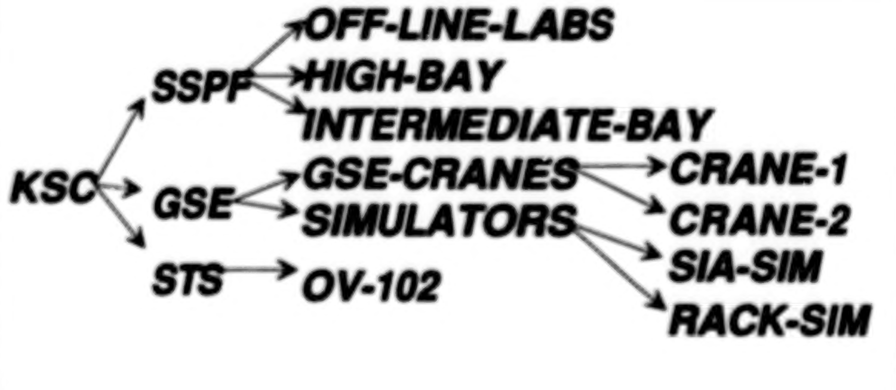


Figure 1. Sample Genealogy

The *knowledge base* resides in the HSKB, or *Hierarchical Segmented Knowledge Base*. The hierarchy comes from the object oriented nature of the system. Small portions of the knowledge base, called *rule sets*, are attached to appropriate objects (the segmentation of the HSKB), and these rule sets can be inherited from parent to child, just as with attributes. This structure differs from the typical expert system knowledge base, where all rules reside in a single unit. The segmentation of the rule sets allows the simulation to reason over only those portions of the knowledge germane to the situation.

The *script-based* simulation utilized by AGAPE allows each object (or for similar processes, groups of objects) to contain its own activity capabilities and requirements. These scripts are built from three basic types of activities; *task* activities, such as servicing or testing, *transport* activities, to move an object from one location to another, and *installation/deintegration* activities, where an object installs itself in or removes itself from another object. The scripts also allow objects to define resources needed for a particular activity, such as technicians and lifting devices for a transport activity. These resource requests also provide a method for the script to interact with the HSKB, allowing certain attribute requirements to replace specific object requests. To move a 13,000 pound payload, for instance, the script may call for a specific crane, or it may require any lifting device with a capability of more than 6.5 tons.

The *simulation* in AGAPE offers an animation feature. The specific processing areas can be viewed during the simulation. As the simulation begins, the modeler may wish to view processing within the SSPF. If certain activities are of interest, like the weight and center-of-gravity test area in the high bay, that object can be selected during the animation, and the display will focus on that particular area.

III KNOWLEDGE ACQUISITION AT KSC

3.1 GROUND PROCESSING AT KSC

KSC's function in the NASA payload community is generally considered to be that of a payload integration and test center, with other centers around the country performing the bulk of the operational and strategic planning. While this is true in many cases, a considerable amount of effort occurs regularly at KSC in creation and development of work plans and procedures for present and future NSTS operations. Space Station Freedom has increased the planning duties at KSC to a large extent, due to the number of unique pieces of flight hardware scheduled to pass through Kennedy's processing facilities. Because the planning for Freedom involves much data manipulation to produce models such as facility and equipment utilizations, it was determined that a knowledge-based simulation environment would provide a valuable resource in planning for many programs.

Payload ground processing at KSC involves a series of complex, tightly controlled assembly and test procedures, as the *launch package* is assembled. (the launch package is the set of payloads for one mission, assembled as they would appear in the orbiter's payload bay. As the payload arrives at the center, it is received and inspected for any shipping damage. The hardware is then taken to an assembly area for build-up to its on-orbit configuration. The payload is tested to assure its functionality, and then verified with the space station hardware it will interface with (rack, APAE, etc). All payloads are tested with space station systems to assure compatibility with such items as power distribution and the Data Management System (DMS). The payloads for a single mission are then assembled into a launch package, and interfaces with the orbiter are verified. The assembled launch package is finally installed into the orbiter and launched. The flow outlined typically lasts from three to eighteen months.

One of the major planning and analysis tasks being performed today at KSC involves the design and review of a new facility, the Space Station Processing Facility (SSPF). The reviews include standard items, such as power outlets and office sizes. Because of the limited resources to process the myriad of Freedom hardware, the review process also includes many utilization studies to assess the impacts of changes in the program and its material.

There are many thousands of single operations required to process any payload at KSC. To adequately model a given process, the proper modeling scope is required to define the precision of the simulation's products. The process of preparing a flight rack to accept a payload can be laid out in its exact detail (hundreds of steps), the major processes can be defined (10-12 activities), or the overall process can be laid out in one to two steps. The more detailed the model, the smaller the scope of the simulation needed to provide an accurate and meaningful representation. If the modeler wishes to simulate the entire 20-mission build-up of the space station, for example, it is obvious to use the very highest level of overview (the fewest steps). This forces the simulation to concentrate on the overall utilization of facilities and equipment, the obvious point of such a large model.

3.2 OUR MODEL - FLIGHT OF-1

In order to more effectively delineate the processes occurring in the knowledge base development for AGAPE, one of the Freedom assembly flights, OF-1 (OutFitting flight 1, number 6 in the series), was chosen to act as a candidate mission. This flight information is currently taken from the August, 1988 Space Station Trial Payload Manifest (TPM), the most accurate set of flight information available for space station planning. This mission will allow a demonstration of the types and quantities of information necessary to support KSC process planning.

Flight OF-1 has many payloads of several types (see Table 1). The station payloads include a hand and eye wash station, a glovebox for utilization by experiments, and payload racks. The user payloads found on the manifest include some science payloads (e.g. SAAX 307X, life science 1.8m centrifuge), technology development payloads (e.g. TDMXF, fluid behavior experiment), and commercial endeavors (COMM 1243, Electroepitaxial crystal growth).

Although the various payloads employ the same basic interfaces to the space station, many require modified or unique attributes to their processing flows due to owners' preferences or singular payload characteristics. As can be seen in the TPM, there are several items bound for the space station on each flight, all with some level of processing occurring at KSC. Multiply this example by the 20 assembly flights necessary to complete the station, and the enormity of the planning and scheduling at KSC becomes apparent.

Another variable has been added to the equation with the creation of the Payload Integration Centers (PICs), centers around the country able to perform some of the space station processing that is KSC's

TABLE 1. MANIFEST OF SPACE STATION FREEDOM FLIGHT OF-1

Manifest Code	Name
STATION SUPPLED EQUIPMENT	
RACK	Payload Racks - 16
USLAB	General Lab Support Equipment
USLAB	PMMS Process Fluids
USLAB	Customer Thermal Control System
USLAB	Commode/Hand & Eye Wash
USLAB	ECLS (Hygiene Water)
USLAB	PMMS Ultrapure Water
USLAB	MPS Glovebox
USER SUPPLIED EQUIPMENT	
Manifest Code	Name
TDMX2411	Advanced Adaptive Control
COMM1255	Commercial Organic & Polymer Processing
LOG	Payload Consumables
COMM1243	Commercial Electroepitaxial Crystal Growth
SAAX401C	Modular Containerless Processing Facility
TDMXF	Two-Phase Fluid Behavior
COMM1230	Commercial Crystals by Vapor Transport
SAAX307X	1.8m Centrifuge, Animal Holding Facility
SAAX401A	Space Station Furnace Facility

responsibility. The impacts from these centers has yet to be fully understood, and KSC is responsible for the validation and acceptance of their processing capabilities. Since PICs are in the early concept stage of development, it behooves KSC modelers to make preparations for acceptance of these new processing facilities.

IV KNOWLEDGE ACQUISITION

Numerous information sources exist at KSC for outlining *current* standard processing flows. Operations and Maintenance Instructions (OMIs) and Work Authorization Documents (WADs) are two of the sources available to the studious KSC knowledge engineer. The difficulty in determining Freedom payload processing characteristics stems from the high rate of change of space station resource information. The on-orbit resource types and locations differ almost from day to day. The challenge for the modeler, as shown in Figure 2, is to find the information, as it exists at any given time, and compile it into a meaningful and realistic model of space station processing.

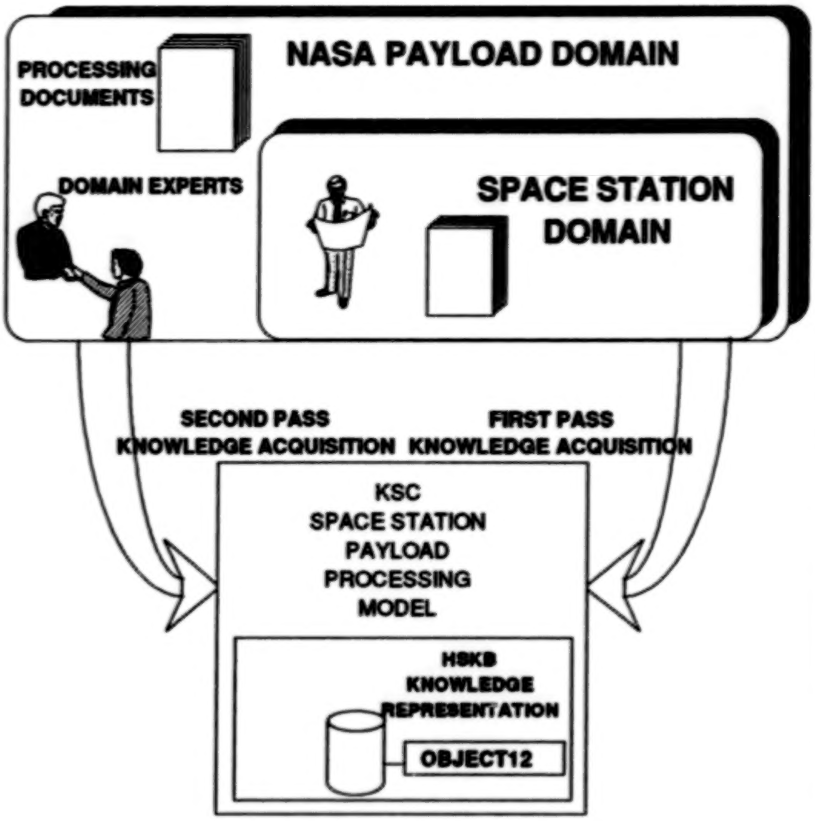

PROCESSING DOCUMENTS
NASA PAYLOAD DOMAIN
DOMAIN EXPERTS
SPACE STATION DOMAIN
SECOND PASS KNOWLEDGE ACQUISITION
FIRST PASS KNOWLEDGE ACQUISITION
KSC SPACE STATION PAYLOAD PROCESSING MODEL
HSKB KNOWLEDGE REPRESENTATION
OBJECT12

Figure 2.
KSC Knowledge Acquisition Process

4.1 ACQUISITION. Because the models being developed represent a program still in its infancy, many of the documents needed to complete the model have not been created. These new space station documents, however, are nearly all based on one or more *existing* documents used to support NSTS processing. For those items in the model requiring a nonexistent document, relevant current documents can be obtained and used as a basis. Generally, once the space station equivalent of the document has received final approval, the changes necessary to update the 'old' model are relatively minor, and quickly implemented and verified. The benefits of developing a model in an object-oriented environment become apparent at this time, as one change propagates through many descendents, making the change almost instantaneous.

At the time of this writing, the SSPF is undergoing tertiary review of its design. As such, specific items like equipment locations and utility port sizes and types are in a nearly constant state of flux. This makes the modeler's job rather difficult, because attribute values are constantly in need of revision. The basic layout of the building, on the other hand, has remained nearly constant over the past few months.

The data for the SSPF comes from two main types of sources, written and personnel. Documents are released in increments as the review process develops, so changes are manifested only when the next document revision is released. Data from the personnel contacts change almost daily while the review process continues. This makes for frustration as a knowledge engineer, since values and concepts are modified from minute to minute, depending on the person being interviewed. Our experience shows that using values from the documentation leads to the fewest problems. Generally, the people involved with the design reviews know when a change in the documents is about to occur, but these alterations should only be incorporated after they have been approved by the appropriate personnel.

Ground processing data and procedures are being formulated at this time. All these documents depend a great deal on the actual configuration of space station hardware, which remains in a state of constant change. The data presented is based on the most recent versions of the information.

4.2 ORGANIZATION. Because of the characteristics of the HSKB utilized by AGAPE, the organization of data easily falls into a familial grouping, much in the same manner as object-oriented programming. For the most part, rules necessary to accomplish a simulation follow the payload type groups: Payloads in general have different scripts than GSE, rack payloads use different rules than attached unpressurized payloads, etc. Similarities in certain aspects of processing allow for use of the same rules, provided that the rules were written in a generic fashion.

4.3 REPRESENTATION. The method by which knowledge is input into the AGAPE system tightly constrains the method of knowledge representation, but in no way limits the modeler in developing a useful and accurate model. Because of the breadth of information necessary to model KSC as a payload processing center, many rules are required for each aspect of payload integration. These rules tend to apply to only one or two types of payloads, or to experiments rather than elements, and thus the knowledge base grows at rather an astounding rate.

The grouping of rule sets is actually controlled in a large manner by the architecture of the system. Unlike traditional expert systems, the reasoning does not involve the entire knowledge base. Instead, as shown in Figure 3, the simulation reasons over only that part of the knowledge attached to the scripts under consideration. This automatic segmentation of the knowledge occurs in such a way that the modeler often fails to notice.

As an example of the representation of the knowledge gathered for the models at KSC, a rule is needed to acquire a crane in the SSPF to relocate a payload as it moves through its processing activities. This rule looks like:

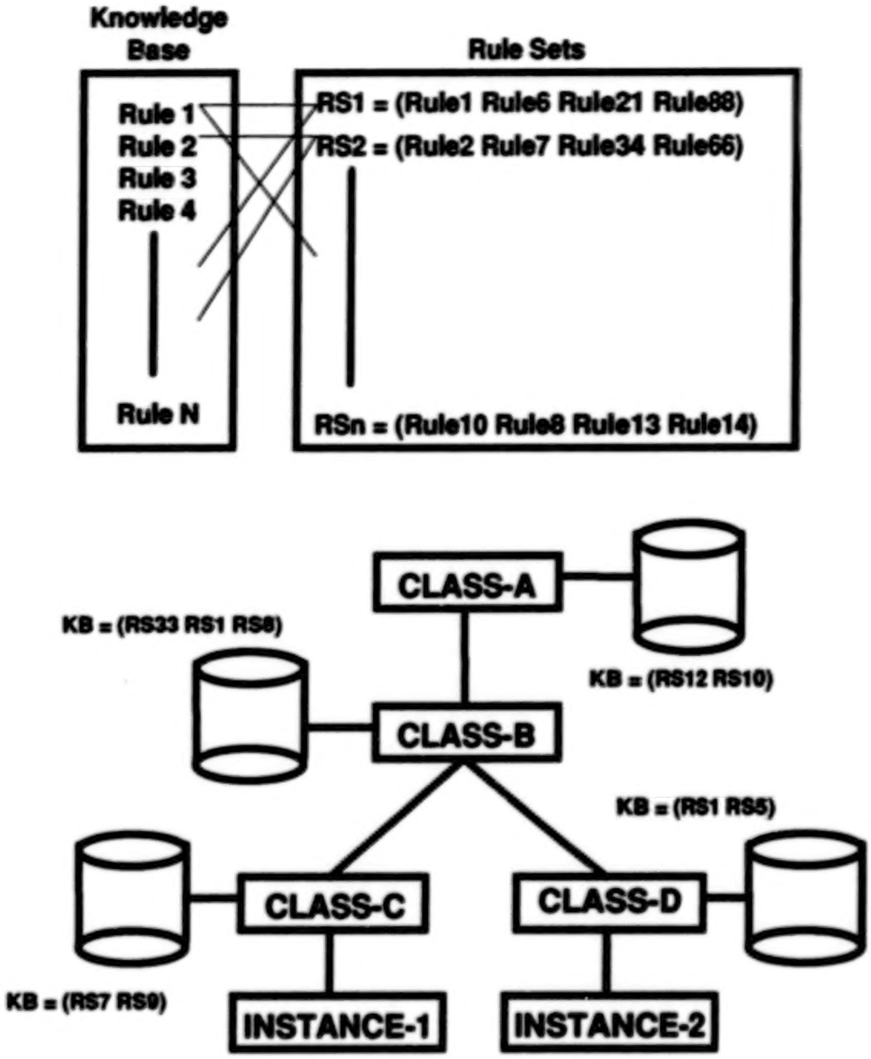

Knowledge Base
Rule Sets
Rule 1
Rule 2
Rule 3
Rule 4
Rule N
RS1 = (Rule1 Rule6 Rule21 Rule88)
RS2 = (Rule2 Rule7 Rule34 Rule66)
RSn = (Rule10 Rule8 Rule13 Rule14)
CLASS-A
KB = (RS33 RS1 RS8)
KB = (RS12 RS10)
CLASS-B
KB = (RS1 RS5)
CLASS-C
CLASS-D
KB = (RS7 RS9)
INSTANCE-1
INSTANCE-2

Figure 3.
Hierarchical Segmented Knowledge Bases

```
(define-hskb-rule P1
   :lhs (ACQUIRE-CRANE ?PAYLOAD ?CRANE
   :rhs ((BIND-IN-LIST ?CRANE (ASK 'GSE-CRANES CHILDREN))
          (> (ASK '?CRANE REQUEST 'LIFT-CAPACITY 'IS)
             (ASK '?PAYLOAD REQUEST 'MASS 'IS))
          (RETURN-VALUE ?CRANE))
```

This rule finds a crane whose lift capacity is simply greater than the mass of the payload. With the backward chaining process used in reasoning, the left-hand side (:lhs) or *consequent* of the rule is true if the right-hand side (:rhs) or *antecedent* is satisfied. All the statements in the rhs of the rule must be satisfied for that side to be true. In the case of this rule, a crane object (variable ?crane) is capable of lifting a payload if the rhs is complete. The BIND-IN-

LIST command assigns the ?crane to each of the children (first-order descendents) of the class GSE-CRANES. Each of the cranes is queried to determine the lift capacity of the device *(ASK '?CRANE REQUEST 'LIFT-CAPACITY 'IS)*. This is compared to the mass of the payload *(ASK '?PAYLOAD REQUEST 'MASS 'IS)* to assure a simply greater-than relation. The first crane to satisfy these requirements is returned to the caller *(RETURN-VALUE ?CRANE)*.

This rule could be expanded to include function calls or calls to other rules, to assure the accuracy of the model. By replacing the greater-than *(>)* line above with:

```
(BIND ?X (ASK '?CRANE REQUEST 'LIFT-CAPACITY 'IS)
(CRANE-CAPACITY (ASK '?PAYLOAD REQUEST 'MASS 'IS) '?X)
```

This section would call a function or a rule *CRANE-CAPACITY* to calculate the proper amount of over-capacity for a crane to safely lift a payload. For example, if KSC requires the lifting device to exceed the payload's mass by 40%, or use a more complicated formula, or tabular values, the rule could accommodate the requirements.

Figure 4 contains a copy of the rule editor screen for AGAPE. The rule set shown checks a rack to assure that a payload will conform to the proper type and dimensions. The pressurized-module-code refers to the on-orbit residence of the rack payload (US Laboratory module, ESA, Japanese).

These rules can be utilized at many points in the simulation. Any time in a script that a certain rule set is needed, an HSKB message can be attached to activate rule usage. To get a crane to relocate a payload within the SSPF, for instance, a message could be used. The text of the message would appear as:

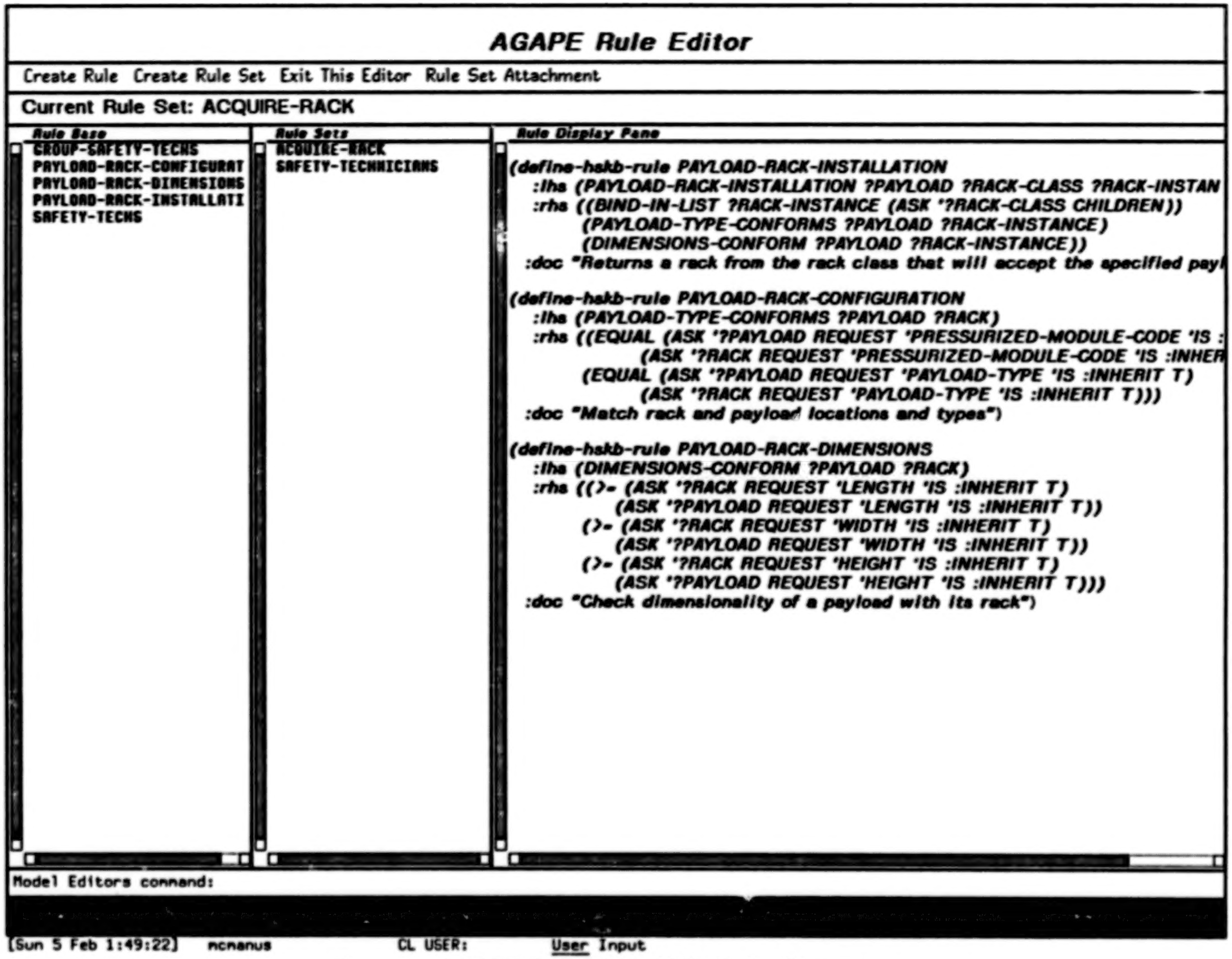
```
(DEFINE-HSKB-MESSAGE 'PAYLOAD 'MESSAGE26
      ()
      (ACQUIRE-CRANE ?SELF ?CRANE))
```

The rule *ACQUIRE-CRANE* is passed the values SELF (the payload calling the message) and CRANE, which will contain the crane returned by the rules. This will allow the script to acquire a crane to perform its current activity.

Figures 5 and 6 display the code which expands to write rules and messages. This code, along with the rest of AGAPE, is public domain software available in COSMO.

V CONCLUSION AND FUTURE RESEARCH

As the models under development at KSC become more robust and flexible, many varied processes could be simulated. The current processing of STS payloads could be studied, along with the processing of the shuttles themselves. KSC's budgetary cycle could also be modeled. Impacts of future NASA programs, such as lunar base concepts



Current Rule Set: ACQUIRE-RACK

```
GROUP-SAFETY-TECHS
PAYLOAD-RACK-CONFIGURAT
PAYLOAD-RACK-DIMENSIONS
PAYLOAD-RACK-INSTALLATI
SAFETY-TECHS
```

```
ACQUIRE-RACK
SAFETY-TECHNICIANS
```

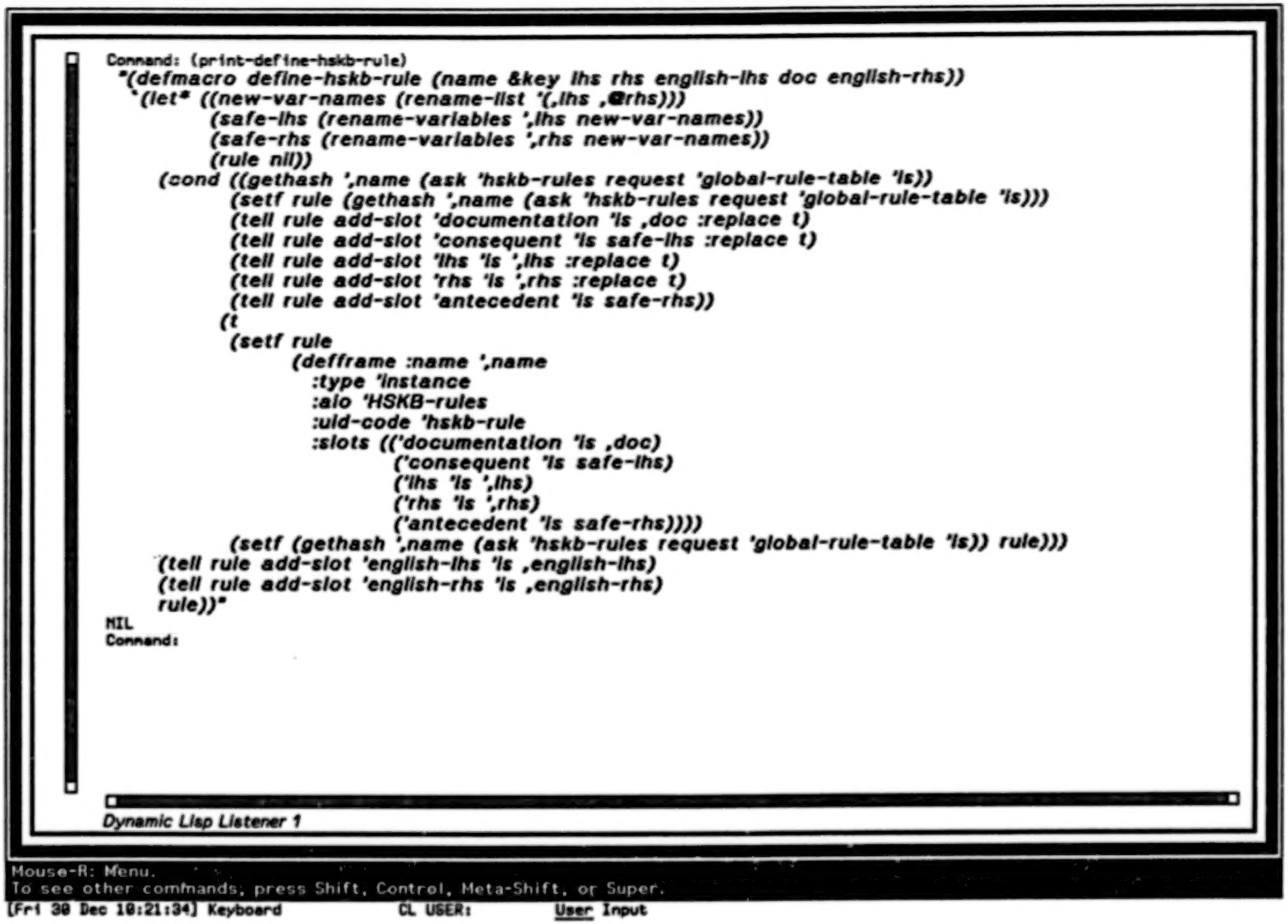
```
(define-hskb-rule PAYLOAD-RACK-INSTALLATION
  :lhs (PAYLOAD-RACK-INSTALLATION ?PAYLOAD ?RACK-CLASS ?RACK-INSTAN
  :rhs ((BIND-IN-LIST ?RACK-INSTANCE (ASK '?RACK-CLASS CHILDREN))
        (PAYLOAD-TYPE-CONFORMS ?PAYLOAD ?RACK-INSTANCE)
        (DIMENSIONS-CONFORM ?PAYLOAD ?RACK-INSTANCE))
 :doc "Returns a rack from the rack class that will accept the specified payl

(define-hskb-rule PAYLOAD-RACK-CONFIGURATION
  :lhs (PAYLOAD-TYPE-CONFORMS ?PAYLOAD ?RACK)
  :rhs ((EQUAL (ASK '?PAYLOAD REQUEST 'PRESSURIZED-MODULE-CODE 'IS :
               (ASK '?RACK REQUEST 'PRESSURIZED-MODULE-CODE 'IS :INHER
        (EQUAL (ASK '?PAYLOAD REQUEST 'PAYLOAD-TYPE 'IS :INHERIT T)
               (ASK '?RACK REQUEST 'PAYLOAD-TYPE 'IS :INHERIT T)))
 :doc "Match rack and payload locations and types")

(define-hskb-rule PAYLOAD-RACK-DIMENSIONS
  :lhs (DIMENSIONS-CONFORM ?PAYLOAD ?RACK)
  :rhs ((>= (ASK '?RACK REQUEST 'LENGTH 'IS :INHERIT T)
            (ASK '?PAYLOAD REQUEST 'LENGTH 'IS :INHERIT T))
        (>= (ASK '?RACK REQUEST 'WIDTH 'IS :INHERIT T)
            (ASK '?PAYLOAD REQUEST 'WIDTH 'IS :INHERIT T))
        (>= (ASK '?RACK REQUEST 'HEIGHT 'IS :INHERIT T)
            (ASK '?PAYLOAD REQUEST 'HEIGHT 'IS :INHERIT T)))
 :doc "Check dimensionality of a payload with its rack")
```

```
Model Editors command:
```

```
[Sun 5 Feb 1:49:22]    mcmanus        CL USER:        User Input
```

Figure 4. AGAPE Rule Editor With Rules Shown

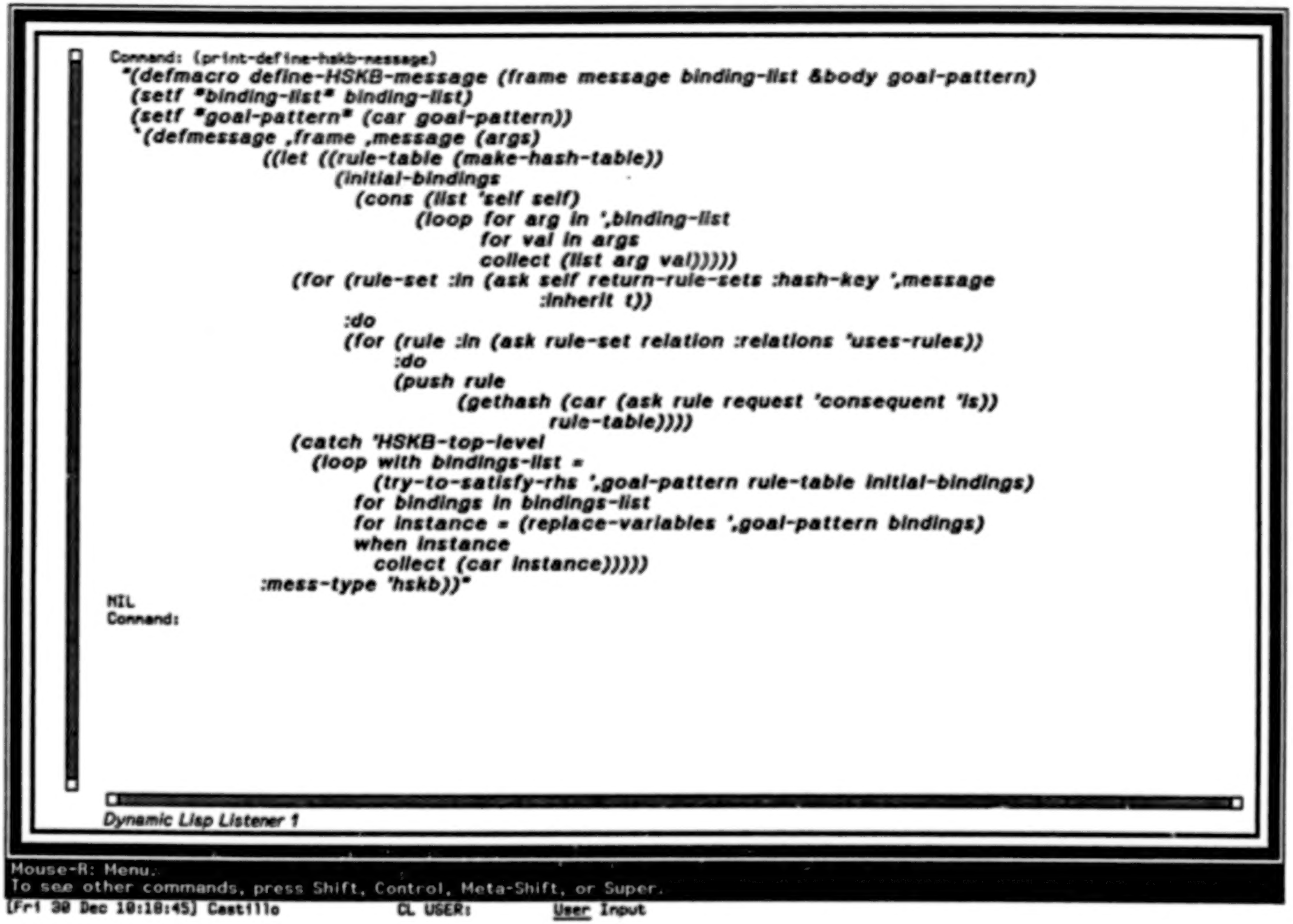
BEST COPY AVAILABLE



```
Command: (print-define-hskb-rule)
 "(defmacro define-hskb-rule (name &key lhs rhs english-lhs doc english-rhs))
  `(let* ((new-var-names (rename-list '(,lhs ,@rhs)))
          (safe-lhs (rename-variables ',lhs new-var-names))
          (safe-rhs (rename-variables ',rhs new-var-names))
          (rule nil))
     (cond ((gethash ',name (ask 'hskb-rules request 'global-rule-table 'is))
            (setf rule (gethash ',name (ask 'hskb-rules request 'global-rule-table 'is)))
            (tell rule add-slot 'documentation 'is ,doc :replace t)
            (tell rule add-slot 'consequent 'is safe-lhs :replace t)
            (tell rule add-slot 'lhs 'is ',lhs :replace t)
            (tell rule add-slot 'rhs 'is ',rhs :replace t)
            (tell rule add-slot 'antecedent 'is safe-rhs))
           (t
            (setf rule
                  (defframe :name ',name
                    :type 'instance
                    :aio 'HSKB-rules
                    :uid-code 'hskb-rule
                    :slots (('documentation 'is ,doc)
                            ('consequent 'is safe-lhs)
                            ('lhs 'is ',lhs)
                            ('rhs 'is ',rhs)
                            ('antecedent 'is safe-rhs))))
            (setf (gethash ',name (ask 'hskb-rules request 'global-rule-table 'is)) rule)))
     (tell rule add-slot 'english-lhs 'is ,english-lhs)
     (tell rule add-slot 'english-rhs 'is ,english-rhs)
     rule))"
NIL
Command:
```

```
Mouse-R: Menu.
To see other commands, press Shift, Control, Meta-Shift, or Super.
[Fri 30 Dec 10:21:34] Keyboard          CL USER:        User Input
```

Figure 5. Expanded Rule Macro



```
Command: (print-define-hskb-message)
 "(defmacro define-HSKB-message (frame message binding-list &body goal-pattern)
  (setf *binding-list* binding-list)
  (setf *goal-pattern* (car goal-pattern))
  `(defmessage ,frame ,message (args)
             ((let ((rule-table (make-hash-table))
                    (initial-bindings
                     (cons (list 'self self)
                           (loop for arg in ',binding-list
                                for val in args
                                collect (list arg val)))))
                (for (rule-set :in (ask self return-rule-sets :hash-key ',message
                                     :inherit t))
                    :do
                    (for (rule :in (ask rule-set relation :relations 'uses-rules))
                         :do
                         (push rule
                              (gethash (car (ask rule request 'consequent 'is))
                                      rule-table))))
                (catch 'HSKB-top-level
                  (loop with bindings-list =
                       (try-to-satisfy-rhs ',goal-pattern rule-table initial-bindings)
                     for bindings in bindings-list
                     for instance = (replace-variables ',goal-pattern bindings)
                     when instance
                       collect (car instance)))))
             :mess-type 'hskb))"
NIL
Command:
```

```
Mouse-R: Menu.
To see other commands, press Shift, Control, Meta-Shift, or Super.
[Fri 30 Dec 10:18:45] Castillo          CL USER:        User Input
```

Figure 6. Expanded Message Macro

and planetary missions, could be assessed to determine long-range goals.

With the user-oriented capabilities of the AGAPE system, coupled with its robustness, focus the process of knowledge engineering into a single, continuous technique. Allowing the models to be developed *directly* by the people performing the payload processing brings the actual knowledge one step closer to the simulation. This system could assist many studies at KSC, not to mention processing scenarios at other NASA centers.

BLANK PAGE

A Heuristic Approach to Incremental and Reactive Scheduling

Jidé B. Odubiyi and David R. Zoch

Ford Aerospace Corporation
Space Systems Engineering Operation
7375 Executive Place
Seabrook, MD 20706

Abstract

This paper describes a heuristic approach to incremental and reactive scheduling. Incremental scheduling is the process of modifying an existing schedule if the initial schedule does not meet its stated initial goals. Modifications made to a schedule during incremental scheduling typically consist of adding one or more activities by re-scheduling existing activities. Reactive scheduling is performed when changes need to be made to an existing schedule due to uncertain or dynamic environments such as changes in available resources or the occurrence of targets of opportunity. Only minor changes are made during both incremental and reactive scheduling because a goal of re-scheduling procedures is to minimally impact the schedule.

A scheduling system generates a schedule in three phases. An initial batch scheduling phase, an incremental scheduling phase and a reactive scheduling phase. During the first phase, no rescheduling is attempted. All user requests are submitted to the scheduler and an initial schedule is created. During the second phase, non-computationally complex strategies must be used since the number of possible schedules that can be generated increases exponentially with the number of requests. Since simple strategies must be used for initial schedule creation, any schedule can potentially be greatly improved through the use of an incremental scheduling phase.

Reactive scheduling occurs in near real-time in response to the occurrence of targets of opportunity. Consequently, a reactive scheduler must be able to generate schedules within acceptable time limits. Manual reactive scheduling is an inefficient strategy, and automated exhaustive search techniques are infeasible because of time limits.

This paper describes the heuristic search techniques employed by the Request Oriented Scheduling Engine (ROSE), a prototype generic scheduler (3). Specifically, we describe heuristics that efficiently approximate the cost of reaching a goal from a given state and effective mechanisms for controlling search.

Introduction

Scheduling the Tracking and Data Relay Satellite System's (TDRSS) communications' events and user preferences present the NASA-GSFC's Network Control Center's personnel with a very complex scheduling problem. The schedulers must deal with limited TDRSS resources, such as antennas, ground equipment and communications bandwidth. In addition to these resource constraints, the scheduling requirements also have user constraints, such as TDRS visibility of user spacecraft, as well as temporal and dynamic (request placement with respect to other scheduled requests) constraints.

A sample request is shown in Figure 1 where a user of the Upper Atmospheric Research Satellite (UARS) requests the NCC to schedule a house-keeping activity for UARS 19 times, once every 80 minutes, and each request must start within a 40 minute time window. In addition, each request must use a single access antenna from TDRS-East for a period of 15 minutes and it should be scheduled when UARS is in view of TDRS-East.

The scheduling of these requests is premised by the fact that any instances of this generic request should be scheduled only if alternate request instances in a generic request which performs UARS house-keeping using TDRS-West, have not been scheduled. The NCC personnel receive and process several hundreds of requests with more complex requirements from several users on a weekly basis. During the space station era users will generate thousands of such requests.

The Request Oriented Scheduling Engine (ROSE) is a generic scheduling software prototype which has successfully demonstrated the scheduling of user requests in the scheduling of scientific instrument operations for the Space Station distributed scheduling environment, and the scheduling of user requests in the NCC environment. The rest of this paper provides a brief description of the ROSE scheduler, incremental and reactive scheduling processes and the implementation of a hybrid search algorithm to speed automated rescheduling activities.

THE PROBLEM

With thousands of requests to schedule, the initial batch scheduling approach does not usually meet the user's scheduling goals. Also the initial schedule is sub-optimal due to the necessity to use simple heuristics. ROSE provides tools to allow the user to do re-scheduling by deleting or moving scheduled requests, adding unscheduled requests, or relaxing requests' constraints manually.

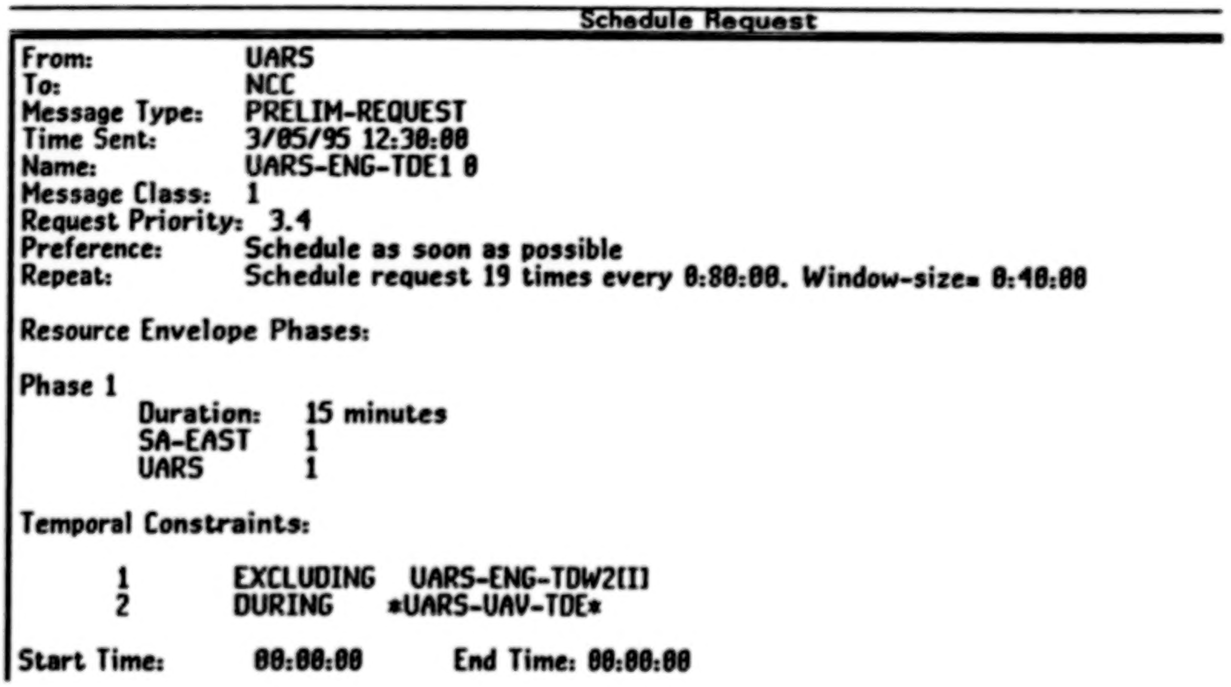
When re-scheduling involves a large number of requests, in order to find a location for an unscheduled request, extensive search of the attributes (i.e., constraints, resources requirements, etc.) of scheduled requests must be performed. This step is required in order to identify appropriate heuristics to improve the search. Also, if the schedule is to be generated in near real-time, the search algorithm must be efficient and fast enough for the resulting schedule to be of any use. Therefore, an automated incremental and reactive scheduling capability is needed in ROSE.

ROSE - A Generic Scheduling Software System

The ROSE software prototype has been developed to provide NASA customers in the Space Station distributed scheduling environment with an automated mechanism for communicating their scheduling requirements to NASA-Goddard Space Flight Center (NASA-GSFC) and receiving their scheduled requests. In ROSE, the feasibility of communicating user requests from remote locations (where appropriate) to a scheduler is being explored. The scheduling requirements are communicated to a NASA scheduler in a Flexible Envelope Request Notation (FERN). This notation enables a user to specify his/her requests with preferential constraints. ROSE/FERN is described in more detail in [3].

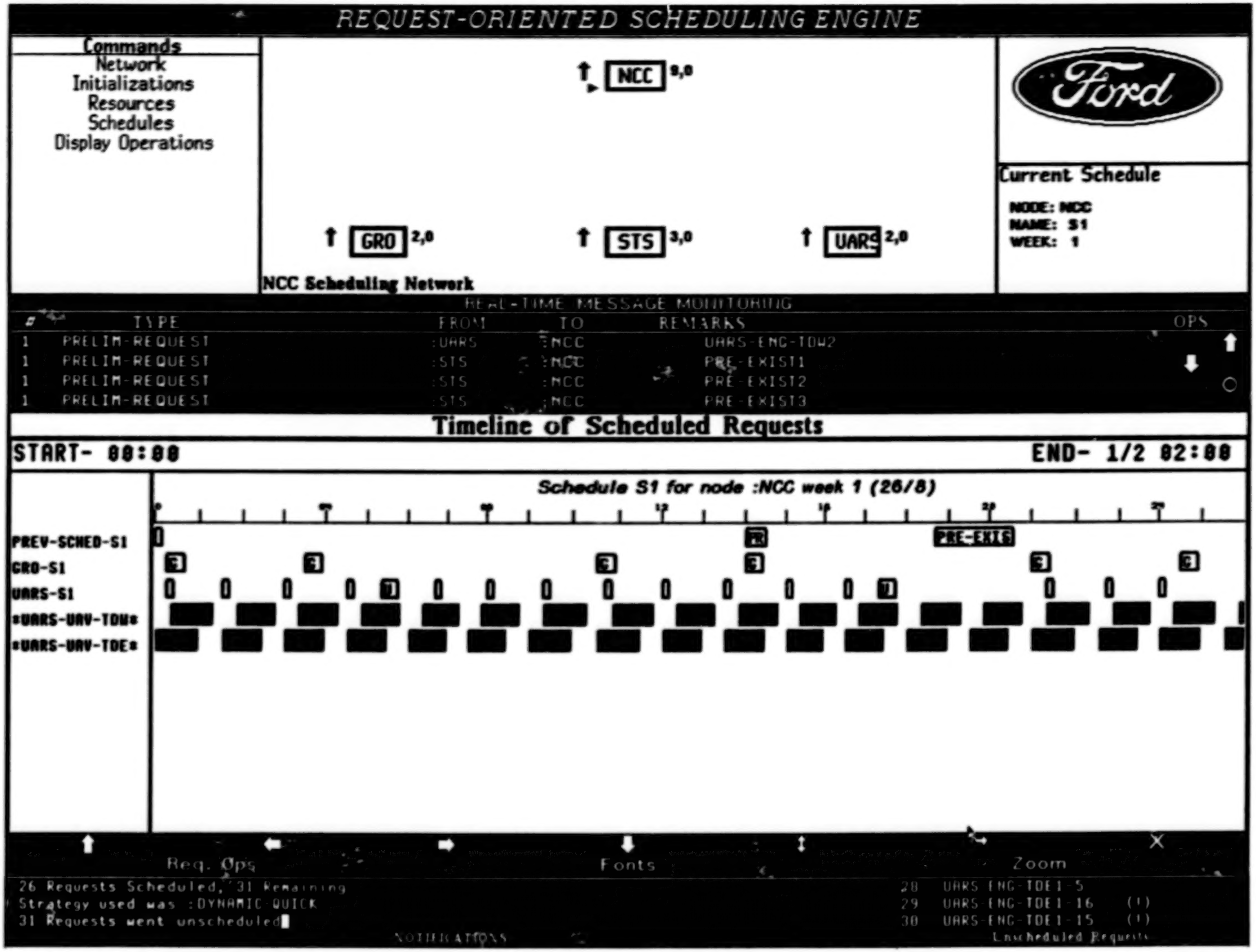
ROSE is a generic scheduler currently running on the Symbolics computer workstation with the Genera 7.0 operating system and the Common LISP language on the Symbolics computer workstation at the NASA-GSFC in code 520. Figure 2 depicts the ROSE user interface. The interface consists of several windows. The user executes many of the ROSE commands by activating the menu items in the Commands window. The NCC scheduling network window shows three users (GRO, STS and UARS) in this example with the NCC as the scheduler. Generic requests from the users to the schedulers are monitored and presented in a scrollable window, titled "Real-Time Message Monitoring".

Figure 2 also shows a day's schedule in the window titled "Timeline of Scheduled Requests". Scheduled requests are displayed as unshaded rectangular boxes along a timeline. The names of the user or campaign are displayed to the left of the corresponding scheduled requests. More information about each scheduled requests can be displayed, and the parameters of the request can be modified through the interface. Below the requests in Figure 2 in shaded rectangles is a sample of TDRS's visibility constraint. The first row of shaded rectangles displays the time windows when UARS is in view of the TDRS-West antenna, while the next row depicts when the same spacecraft is in view of TDRS-East antenna. The bottom right corner window displays a list of unscheduled requests. The window is scrollable, and the user has the option to scroll the window for a list of other unscheduled requests. The user can also mouse each request to obtain a detailed information about the request. ROSE has many features that enable the user to display information about schedules, requests and resource usage.



```
                                        Schedule Request
From:              UARS
To:                NCC
Message Type:      PRELIM-REQUEST
Time Sent:         3/05/95 12:30:00
Name:              UARS-ENG-TDE1 0
Message Class:     1
Request Priority:   3.4
Preference:        Schedule as soon as possible
Repeat:            Schedule request 19 times every 0:80:00. Window-size= 0:40:00

Resource Envelope Phases:

Phase 1
          Duration:   15 minutes
          SA-EAST     1
          UARS        1

Temporal Constraints:

        1         EXCLUDING   UARS-ENG-TDW2[I]
        2         DURING    *UARS-UAV-TDE*

Start Time:        00:00:00        End Time: 00:00:00
```

FIGURE 1. A Sample User's Generic Request


REQUEST-ORIENTED SCHEDULING ENGINE
Commands
Network
Initializations
Resources
Schedules
Display Operations
NCC 9,0
GRO 2,0
STS 3,0
UARS 2,0
NCC Scheduling Network
Ford
Current Schedule
NODE: NCC
NAME: S1
WEEK: 1
REAL-TIME MESSAGE MONITORING
OPS
Timeline of Scheduled Requests
START- 00:00
END- 1/2 02:00
Schedule S1 for node :NCC week 1 (26/8)
0
4
8
12
16
20
24
PREV-SCHED-S1
GRO-S1
UARS-S1
UARS-UAV-TDW
UARS-UAV-TDE
PR
PRE-EXIS
Req. Ops
Fonts
Zoom
NOTIFICATIONS
Unscheduled Requests

#	TYPE	FROM	TO	REMARKS
1	PRELIM-REQUEST	:UARS	:NCC	UARS-ENG-TDW2
1	PRELIM-REQUEST	:STS	:NCC	PRE-EXIST1
1	PRELIM-REQUEST	:STS	:NCC	PRE-EXIST2
1	PRELIM-REQUEST	:STS	:NCC	PRE-EXIST3

```
26 Requests Scheduled, 31 Remaining
Strategy used was :DYNAMIC-QUICK
31 Requests went unscheduled
```

```
28   UARS-ENG-TDE1-5
29   UARS-ENG-TDE1-16    (!)
30   UARS-ENG-TDE1-15    (!)
```

FIGURE 2. ROSE User Interface

Re-Scheduling Strategies to Meet Scheduling Goals

ROSE is a generic scheduler, and it has been developed so that the user can generate schedules with different scheduling goals. When the initial schedule does not meet its goal, the scheduling software, (i.e. ROSE) may take one or more of the following conflict resolution strategies.

- Relax the requirements of unscheduled requests

- Overbook certain resources

- Relax the requirements of scheduled requests or de-allocate certain resources

- Acquire additional resources from another scheduler in a distributed scheduling architecture

- Implement an Incremental Scheduling strategy

- Implement a Reactive Scheduling strategy which incorporates one or more of the courses of action above

In this paper, we only describe the incremental and reactive scheduling strategies for re-scheduling.

Scheduling Goals

A user's scheduling goals can take several forms, for instance:

- Create a schedule within the time limit of T hours.

- Schedule all requests above priority N

- Reserve X% of resource R during time T1.....T2

- Schedule as many requests as possible

With the scheduling goal(s) identified, the ROSE software generates a plan as to which actions to take and in what order, and attempts to generate a schedule that meets the user's goal(s). For example, if the user's goal is to schedule as many requests as possible, the plan may include a step to relax the resource requirements of all requests. Figure 3 depicts a process flow chart used in ROSE for incremental scheduling. ROSE applies each strategy in the plan to the initial schedule until either the user's goals are met or the plan is exhausted.

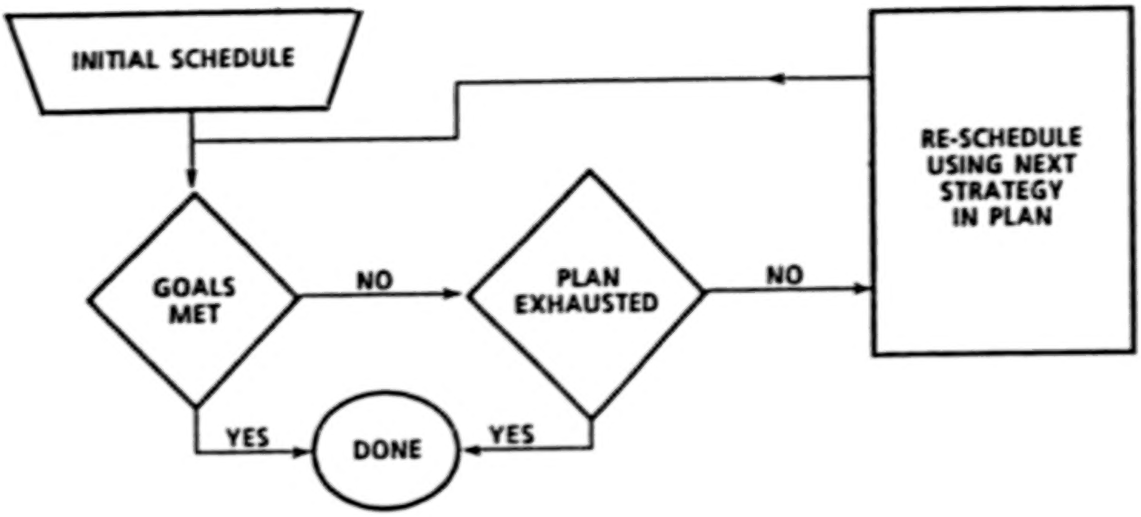


FIGURE 3. A Process Flow Chart for an Incremental Scheduling Strategy

Reactive Scheduling

When an unexpected occurrence of some events triggers the need for replanning, ROSE provides the capabilities to apply the opportunistic scheduling procedure diagrammed in Figure 4. Re-scheduling is performed by adding, moving or deleting requests until the effects of the impacts are eliminated.

Reactive scheduling is used to modify a schedule already in use. Therefore, conflict resolution strategies which are valid in incremental scheduling may not be applicable for reactive scheduling. For example, if a week's schedule already in use requires reactive scheduling at mid-week (i.e. Wednesday), then any requests prior to Wednesday cannot be moved. In other words, anything in the past cannot be moved, and no requests can be scheduled prior to Wednesday. In incremental scheduling, the scheduling system focusses its attention on re-scheduling existing requests in order to accommodate additional requests. In reactive scheduling, however, the scheduler must consider alternative strategies, such as relaxing the requirements of requests in order to minimally impact the schedule. Still, the goal in reactive scheduling is to minimally impact the rest of the schedule.

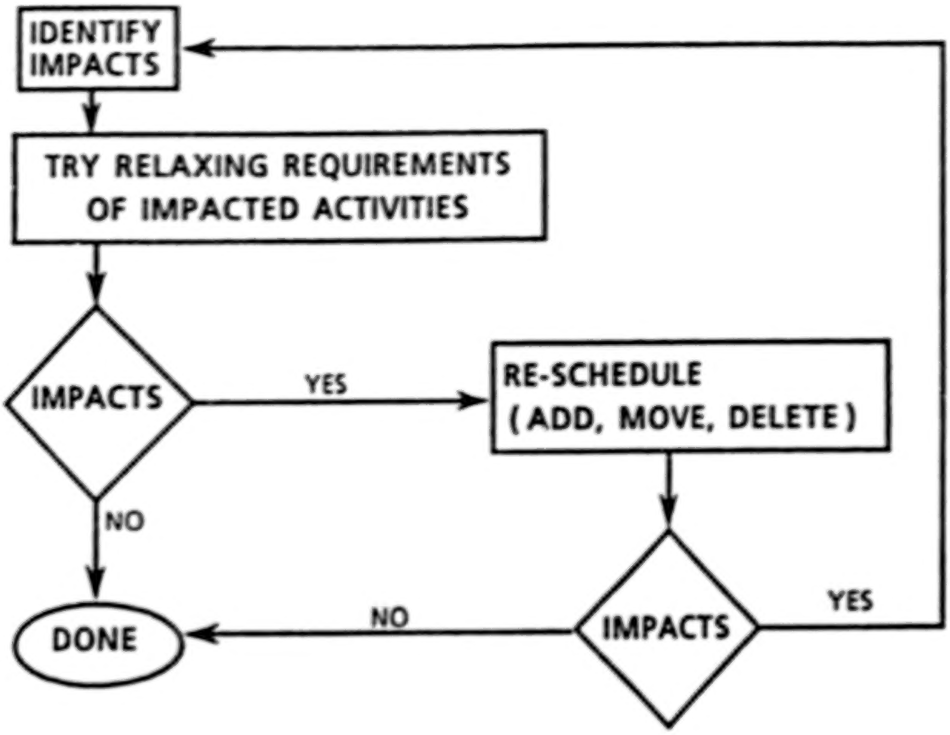

IDENTIFY IMPACTS
TRY RELAXING REQUIREMENTS OF IMPACTED ACTIVITIES
IMPACTS
YES
RE-SCHEDULE (ADD, MOVE, DELETE)
NO
DONE
NO
IMPACTS
YES

FIGURE 4. A Process Flow Chart for a Reactive Scheduling Strategy

Heuristics for Efficient Re-Scheduling

In the field of Artificial Intelligence, several researchers have focussed on developing efficient search techniques for complex mathematically intractable problems. Simon (1962) proposed the "Hierarchical" approach during search by planning at different levels of abstraction. In 1975, Sacerdoti proposed the "Least Commitment" approach which suggests delaying any decision making as much as possible until most of the facts are known and thereby reducing the amount of backtracking. In 1979, Hayes-Roth proposed the "Opportunistic Reasoning" approach by focussing search in highly constrained areas or areas of highest certainty. Dependency directed backtracking is another popular approach employed in searching to reduce the search space of states. Mark S. Fox (7) research efforts on constraint-directed reasoning provides several approaches to reducing the amount of search required in planning. This paper applies a hybrid approach by combining the generate and test problem solving method and the A* algorithm to search the problem space for a solution to a re-scheduling problem.

Implementation of a Heuristic for Efficient Re-Scheduling

We have implemented a hybrid algorithm similar to the A* (Best-first search) algorithm to provide effective search during the re-scheduling process.

Figure 5 shows a directed graph of the search space for re-scheduling in the ROSE scheduling software system. The problem space is developed from the steps involved in re-scheduling in ROSE as described earlier for Figure 2.

Our algorithm searches a directed graph in which each node represents a state in the problem space. It is used to find a minimal-cost overall path or any other path as quickly as possible. In ROSE, the initial state is the initial batch schedule and a request to be scheduled; a goal state is reached when the unscheduled request is scheduled, and no existing request violates any of its resource requirements or temporal or dynamic constraints. The intermediate states consists of the possible states between the initial state and a goal state.

To accomplish the objective of going from the initial state to the goal state in Figure 5, we employ the generate and test problem solving strategy to generate the rules to guide possible moves. These rules are described in the steps below:

Step 1. Start at level 0 and select an unscheduled request.

Step 2. Generate a set of start times and assign ratings to how good the possible locations where the request can be scheduled are. Good locations are those where the minimum number of constraints are violated and the minimum number and amount of resources are required.

Step 3. Schedule this request in a location where it is constrained the least, either by a resource or a dynamic constraint. Break ties by selecting the location with the earliest time along the timeline. A location with a missing resource is preferred over another location with a violated dynamic constraint. A temporal constraint must not be violated. This step will usually invalidate the current schedule.

Step 4. Create a window around all the requests overlapped by the current request, and identify any such requests as possible candidates to be moved. The local goal is to move one or more requests and re-schedule them elsewhere to make the schedule within this window valid.

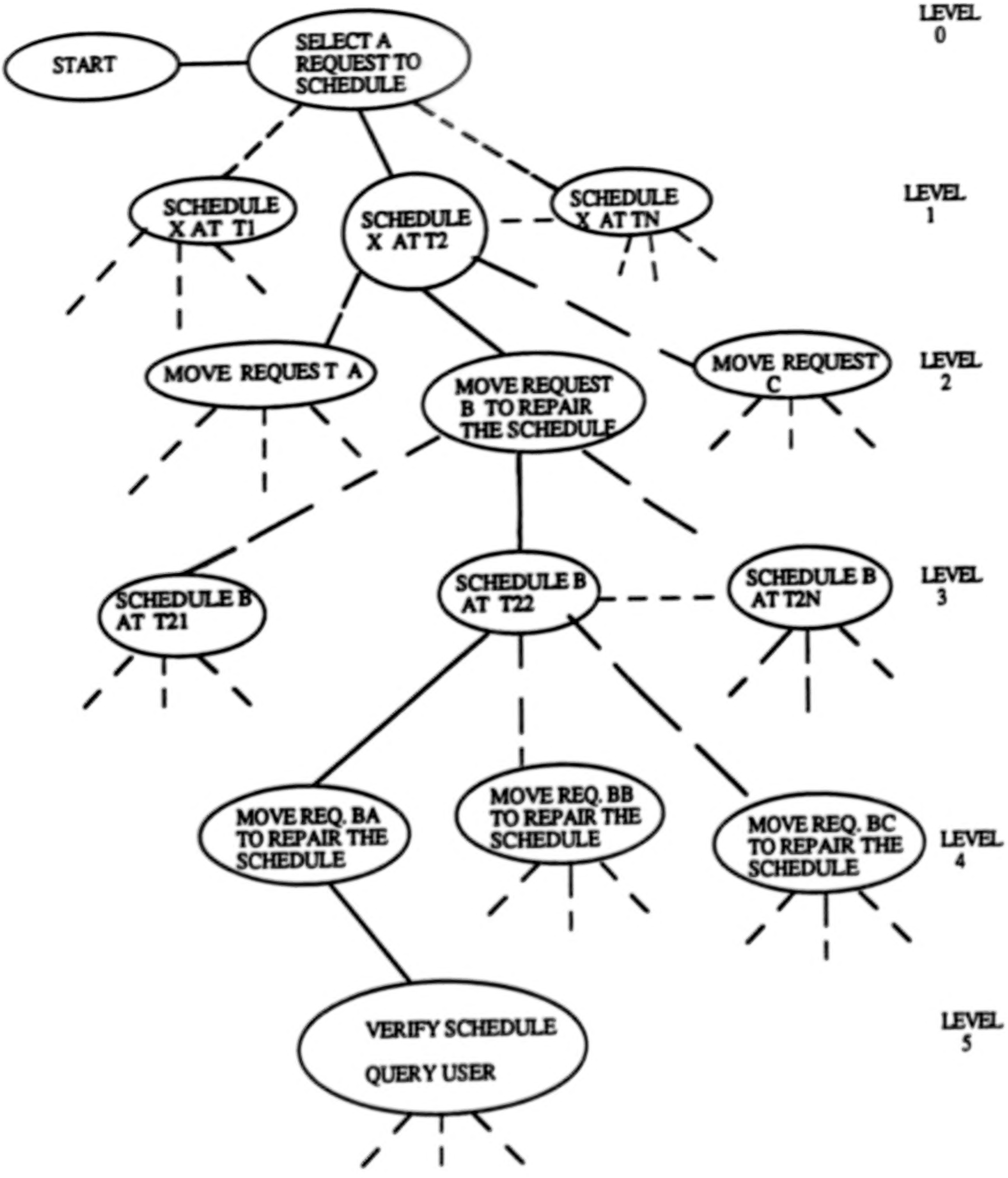

START
SELECT A REQUEST TO SCHEDULE
LEVEL 0
SCHEDULE X AT T1
SCHEDULE X AT T2
SCHEDULE X AT TN
LEVEL 1
MOVE REQUEST A
MOVE REQUEST B TO REPAIR THE SCHEDULE
MOVE REQUEST C
LEVEL 2
SCHEDULE B AT T21
SCHEDULE B AT T22
SCHEDULE B AT T2N
LEVEL 3
MOVE REQ. BA TO REPAIR THE SCHEDULE
MOVE REQ. BB TO REPAIR THE SCHEDULE
MOVE REQ. BC TO REPAIR THE SCHEDULE
LEVEL 4
VERIFY SCHEDULE
QUERY USER
LEVEL 5

FIGURE 5. A SEARCH SPACE FOR RE-SCHEDULING IN ROSE

Step 5. Move an overlapped request within the window. Go to Step 2. Avoid generating any loops by not moving a request more than once.

Step 6. Determine the goodness of a schedule by tallying and evaluating the amount and number of overused resources and the number of dynamic constraints violated by the unscheduled requests

Step 7. Terminate the search at a pre-set time corresponding to the time it takes to reach a certain number of branching and/or depth factor, or until a solution is reached.

Step 8. Display the "best" solution and ask the user if he/she wishes to continue.

Step 9. If the user wishes to continue, attempt to schedule the remaining unscheduled requests.

After establishing the rules that guide acceptable procedures for rescheduling, we are ready to apply our algorithm. To apply this algorithm, we develop an evaluation function, f' which estimates the relative merit or cost of continuing a search from a given state after applying a rule. The evaluation function is a cost function which must be designed to estimate the remaining length of a path between a node n and the goal node. It is used to set up the order as to which nodes to consider during a search such that the goal is reached with the minimum number of steps.

Application of the A* Algorithm

The problem space consists of nodes (shown in Figure 5), and these nodes fall into two categories: OPEN and CLOSED. OPEN is a list of nodes containing the nodes to which the heuristic evaluation function have been applied, but for which their successor nodes have not been generated. The nodes in the list are sorted in a priority sequence such that the highest priority is assigned to the node for which the value returned by the heuristic evaluation function is most promising. The CLOSED list contains the nodes with non-promising values for the evaluation function.

Function f' has two components, a g component and an h' component.

f'(successor node) = g(successor node) + cost to new node
or f'(successor node) = g(successor node) + h'(successor node)

where

g(successor node) = g(best node) + h'(successor node)

and

a best node is defined as a node on OPEN list of nodes with the lowest f'.

The g component is defined as the measure of the cost of getting from the initial state to the current state. It is the sum of the costs of applying the evaluation function along the best path leading to the current node. Function h' returns an estimate of the additional cost of getting to the goal node from the current node. Since h' represents cost, low values for h' lead to good nodes. Implementing the functions described above enables the rescheduling functions to search and reach the goal by manipulating the list of nodes in the OPEN and CLOSED lists.

Since the only action taken at each step is to re-schedule an existing request, the cost of going from one node to its successor node (h') is a constant. If different actions were taken at different nodes (for instance, relaxation and deletions), the h' function will not be constant.

Another Approach to Speed Search During Re-scheduling

The AO* or the AND/OR graph can be used to represent search strategies by decomposing a problem into subproblems. This allows for the generation of alternative solutions to the problem. The initial problem corresponds to the root node of the graph. At an AND node, all the successor nodes must be solved to obtain a solution for that AND node. However, at an OR node, only one of the children nodes must be solved. It is not necessary to generate a solution for more than one node.

Applying this problem solving strategy to searching the search space in Figure 5, it means that in locations where more than one scheduled request must be moved, all the scheduled requests that need to be moved must be moved in parallel until the schedule in a local region becomes valid. This action requires more knowledge of multiple requests. With more knowledge of each requests moved, the amount of search required is reduced, and solution can be obtained at a smaller cost than with the A* algorithm.

Future Work in Automated Re-Scheduling for ROSE

In the future, we plan to explore the application of assumption-based or justification-based truth maintenance system concepts to evaluate their effectiveness in helping to repair invalid schedules generated during re-scheduling. Also, due to the extensive amount of search required, and since we want to limit back-tracking while the scheduler is in search of a goal, some machine learning paradigms such as, learning from experience can speed the re-scheduling time. Also the effectiveness of the application of neural network algorithms in re-scheduling will be explored.

The effects automating other conflict resolution strategies, such as overbooking certain resources, acquiring additional resources from other schedulers in a distributed scheduling architecture will be employed.

CONCLUSION

Quoting Raj Reddy's [6], fourth and fifth rules of Artificial Intelligence, "Search compensates for lack of knowledge" and "Knowledge eliminates the need for search", these statements apply in many problem solving efforts, specifically when solving planning problems. The amount of search required in the heuristic described above can be reduced significantly with more knowledge of the constraints. With a better knowledge of the constraints, the AO* search heuristic can provide a faster solution and a shorter path search than the technique described in this paper. According to Mark Fox (7), scheduling is not yet a science, it is still an art. As a result efficient problem solving techniques must be explored to improve search and reduce re-scheduling time. This paper presents our attempt at improving the time required for automatic re-scheduling in the Space Station and the TDRSS Network Control Center environment. We employed a hybrid problem solving technique to reduce automated re-scheduling time. Given the knowledge of the problem space, the hybrid problem solving approach described here is efficient for re-scheduling.

Bibliography

1. Dean, Thomas, Planning Paradigms, Brown University, Department of Computer Science, Providence, R.I. 02912.

2. Rich, Elaine, Artificial Intelligence, The University of Texas at Austin, McGraw-Hill Book Company, N.Y., 1983.

3. Zoch, David and Hall, Gardiner, 1988, Integrated Resource Scheduling In A Distributed Environment, Ford Aerospace Corporation, 7375 Executive Place, Seabrook, MD 20706.

4. Tanimoto, Steven L. The Elements of Artificial Intelligence, Department of Computer Science, FR-35, University of Washington, Seattle Washington, Computer Science Press, 1987.

5. Dougherty, E. R., and Giardina, C. R., Mathematical Methods for Artificial Intelligence and Autonomous Systems, Prentice Hall, Englewood Cliffs, New Jersey 07632, 1988.

6. Reddy, Raj, (1988), Foundations and Grand Challenges of Artificial Intelligence, AI Magazine, Winter 1988, pp. 9-21.

7. Fox, S. M., (1987), Constraint Directed Search: A Case Study of Job-Shop Scheduling, Morgan Kaufmann Publishers, Inc. Los Altos, California.

8. Minasi, Mark, (1989), A Final Look at Simple Search, AI Expert, February 1989, pp. 15-20.

9. GSFC Code 522, Software and Automated Systems Branch, User's Guide for the Flexible Envelope Request Notation (FERN), Schedule Integration Requirements Study, SEAS Task 29-1000, January 1989

10. GSFC Code 522, Software and Automated Systems Branch, User's Guide for the Request Oriented Scheduling Engine (ROSE), Integrated Resource Scheduling (IRS), SEAS Task 29-1000, January 1989

11. GSFC Code 522, Software and Automated Systems Branch, Schedule Integration Requirements Study, Final Report, SEAS Task 29-1000 January 1989.

Acknowledgements

We would like to thank Nancy Goodman, Larry Hull, Mike Tong and other staff members of NASA-GSFC Code 522 for their support in the development of ROSE.

Fault Isolation/Diagnosis

BLANK PAGE

A METHOD FOR INTERACTIVE SATELLITE FAILURE DIAGNOSIS: TOWARDS A CONNECTIONIST SOLUTION

P.Bourret [+*] J.A.Reggia [*]

+ONERA-CERT
2 Av. E.Belin
31055 Toulouse CEDEX
FRANCE

*UNIVERSITY OF MARYLAND
Department of Computer Science
College Park MD 20742
USA

ABSTRACT

In this paper we briefly analyze the various kind of processes which allow one to make a diagnosis.Then we focus on one of these processes used for satellite failure diagnosis.This process consists of sending instructions to the satellite about system status alterations to make masked the effects of one possible component failure or to look for additional abnormal measures.

A formal modele of this process is given.This model is an extension of a previously defined connectionist model which allows computation of ratios between the likelihoods of observed manifestations according to various diagnostic hypothteses.We show that we are able to compute in a similar way the expected mean value of these likelihood measures for each possible status of the satellite.Therefore,we are able to select the most appropriate status according to three different purposes:to confirm an hypothesis,to eliminate an hypothesis,or to choose between two hypotheses.

Finally a first connectionist schema of computation of these expected mean values is given

I Introduction

There are a lot of human activities which involve diagnostic problem solution.This kind of problem solving typically calls to the mind the physician activity looking for the diseases which may be the cause of observed symptoms.However similar mental processes are involved when a detective looks for a murderer,various specialists try to repair a device and when a scientist tries to dtermine the composition of a given sample(proteins by electrophorese for a biologist,chemical composition by spectrum analysis in chemistry, etc...

1.1.1 Various diagnosis procedures

Everyone who makes a diagnosis does not uses exactly the same reasoning process.How one infers a diagnosis depends in part upon their way to get information.We can distinguish at least three main classes:1) All needed information is immediatly available

2) A part of the needed information is masked from the person trying to make a diagnosis

3) The cause of the observed symptoms may change during the gathering of information. This last case is the most difficult and we will only consider the first two kinds of diagnosis here.

1.1.2 The most general procedure

The basic information with which one can deal in a diagnostic process consists of propositions like:a given disorder may cause a given symptom or manifestation.Therefore,diagnostic inference cannot only be a deductive process:some symptoms of a disorder may be absent and symptoms can be the consequence of several disorders.The mathematical modelling of the cause has been the aim of several previously published papers:[PEAR86],[PEAR87],[BOUR87],[WALD89].For some of them "may cause" is represented by a probability,[PEAR86],[PEAR87],and for others there is a numerical "causal strength" between a disorder and its manifestations,[PENG87].In still another model,[BOUR87] the observed manifestations may also be caused by unknown disorders and a measure of likelihood of each manifestation is introduced in order to be able to neglect the lessprobable manifestations when the deductive process leads to contradictions

1.1.3 The stepwise procedures

Inpractice the diagnostic process consists of two alternate phases:

First seach for a set of plausible hypotheses which explain the set of obseved manifestations, and

Second,confirmation and/or elimination of selected hypotheses.

In order to confirm or to eliminate an hypothesis one can proceeds in two ways

a)New queries The simplest approach is to ask new queries the results of which would enable us to confirm or to eliminate an hypothesis.But such an approach implies that all needed information is available.This is not true in many cases.For instance in a satellite the measured information is chosen when designing the satellite and cannot be changed when the satellite is in space.

b)The indirect diagnosis procedure This second process is applied when a diagnosis is needed for a still working device (satellite,in flight aircraft,boats,etc...).This device has many possible working modes.Each working mode may mask the effects of some disorders.Therefore by change of working mode (within the limits of posible working mode at a given time),people doing diagnosis are able to confirm or to eliminate a given hypothesis.For instance, in a satellite failure diagnosis the operator may force the battery to supply power to various components in order to eliminate the assumption "solar cells failure",if the manifestations diapear with this new working mode.

1.2 The Satellite Failures Diagnosis Procedure

The indirect diagnosis procedure has already been studied[BOUR86].It can be summarized as follows.When an alarm is on in a satellite control room,contollers first apply the emergency procedure related to this alarm.They,then,try to analyse the latest information which has been sent by the satellite to determine if there is a failure and, if so, what kind of failure is it.Because the emegency procedure always protect the satellite,sevral hours can be used to make an accurate diagnosis.In the case of low level satellites,it is not possible to try several working modes within a revolution because the satellite can only received one command and send information back during the short period in which it is visible from antennae.Minimizing

BEST COPY AVAILABLE

the number of needed working modes to complete the diagnosis is thus of prime importance in this case. The deduction process is the following:controllers have at their disposal schemas with various levels of details.They start from the measured point of the sschema which has caused the alarm and follow the functional links starting from one component that arrive at this point.Then they make the assumption that there is a breakdown in this component.After that they try to eliminate this assumption by looking for information,among that most recently received,which contradicts their assumption.If they one they follow back the functional links until they identify a new component.This process seems so simple that one might think that advanced information systems are not required. But, in fact the process is made more difficult by two things.First,in every satellite there are a lot of automatic reconfigurations that occur in order to avoid hazardous effects of a failure.Thus the controller has at his/her disposal only a few pieces of information about what has really happened.More often he/she only knows that an automatic reconfiguration has happened on a given device.He/she must look through long sequences of measurements in order to detect what part of the device has broken.Second,in most cases information is gathered on board the satellite and periodically sent to the control center without information on the time at which each has been gathered.Only the order in which each piece of information is gathered is known.So it is sometime very difficult to exactly know when the failure which caused the alarm happened,and thus which information is related to the period before the failure and which one is related to the period after the failure. We can also say that the failures are rare on board a satellite,so controllers are not well trained to face this kind of event.Morover,failures usually occur more frquently at the end of the satellite life (typically 7 years).By this time, the designers of the satellite,who are the most qualify to find the failure,are either no longer available or have forgotten a large part of their knowledge about the satellite.Therefore an intelligent decision aid for controllers is absolutely needed.

1.3 The study purposes

In a previous study[BOUR86], an expert system was prototyped to make diagnosis automatically.But the solution had two main drawbacks:it was time consuming (a first list of possible failures needed up to tweenty minutes on a SUN/50) and it did not give any advice for selecting a working mode that would be the most appropriate to confirm or eliminate an hypothesis on the list.Another previous study has shown that making the list of most probable hypothesis can be done using a connectionist model[PENG89].We have wanted to go further and to compute, in a similar way, which working mode of the satellite would give the most information in order to reduce the hypotheses list.

II General Mathematical Model

2.1 Notations and basic assumptions

Let $D=\{d_1,\ldots,d_n\}$ be the set of possible disorders,

$M=\{m_1,\ldots,m_k\}$ the set of manifestations,

$p_i, i=1,\ldots n$ the apriori probabilities of d_i, and

c_{il} the frequencies with wich d_i *causes* m_l ($c_{il}=0$ if there is no causal relation between d_i *and* m_l)

Note that $c_{il}=P(m_l \mid d_i)$.For detailled explanations of this point see[PENG88]

Let $C=\{c_{il}\}$ and let $e(d_i)=(m_l : c_{il} \neq 0)$ Let be M^+ the set of observed manifestations in the current working mode W_0 and $M^-=M-M^+$ Let $D_I \subset D$ be an assumption representing a set of possible disordres which can explain all observed manifestations M^+ The following three assumptions are made:

1)Disorders are independent of each other

2)Causal strength (c_{il}) are invariant :whenever d_i occurs it always causes m_l with the same strength.

3)No manifestations can be present without being caused by some disorder.

Define the **Relative Likelihood** measure of $D_I \subseteq D$,given M^+, to be $L(D_I \mid M^+)=P(M^+ \mid D_I) \prod_{d_i \in D_I} \frac{p_i}{1-p_i}$

Where $P(M^+ \mid D_I)$ stands for the probabilitybof the observed set of manifestations, given the set of disorders D_I.

BEST COPY AVAILABLE

II.2 Mains results Let $\alpha_i = \prod_{m_j \in e(d_i) - M^+} (1-c_{ij} \frac{)p_i}{(1-p_i)}$ [1]

Let $L_1(D_I, M^+) = \prod_{m_j \in M^+} (1- \prod_{d_i \in D_I} (1-c_{ij}))$ [2]

Then $L(D_I, M^+) = L_1(D_I, M^+) \prod_{d_i \in D_I} \alpha_i$ [3]

$= L_1(D_I, M^{+)UB(D_I}, m^+)$ where $UB(D_I, M^+) = \prod_{d_i \in D^I} \alpha_i$

Defition of a "confort measure" CM

CM is a real number btween 0 and 1 which represents how certain we wish to be that a collection of diagnosis hypotheses $(D_1, D_2, \ldots, D_k)$ includes the actual set of causitives disorders that are present.

Definition of a minimal solution of a diagnosis problem

Let D,M,C,M^+ be a diagnosis problem that we wish to solve given a confort measure CM $(0 \leq CM \leq 1)$.S$=\{D_1, D_2, \ldots, D_k\} \subseteq$ "subsets of D" is said to be a minimal solution the problem iff

1)P$(D_1 \bigcup D_2 \cdots \bigcup d_k \mid M^+) = \sum_{i=1}^{k} P(D_i \mid M^+) \geq CM$

2) for all $D_j \in S$ $\sum_{i=1, i \neq j}^{k} P(D_i \mid M^+) \leq CM$

Let $A_{D_I} = \sum_{d_i \in D \ D_I} \alpha_i$

Theorem 1 [Peng88]

For any hypothesis $D_I \subseteq$ D: $\sum_{D_J \supseteq D_I} L(D_J \mid M^+) \leq UB(D_I, M^+)(e^{A_{D_I}}-1)$ There is an algoritm [PENG88] which allows to determine the k most probable hypothesis among all members of subsets of D and to order them by decreasing probabilities. An hypothesis is said to cover a problem if this hypothesis canexplain all observed manifestations M^+.

Theorem 2 [Peng88]

Let $D_1, D_2, \ldots, D_k$ be the k most probale covers of a problem PB=D,M,C,M^+ where D_k is the least probable among the k covers.Let CM be a given confort measure.Then S$=(D_1, D_2, \ldots, D_k)$ is a solution for problem PB if:

$$\sum_{i=1}^{k} inf\ (D_I) \geq CM \geq \sum_{i=1}^{k-1} sup\ (D_I)$$

where inf$(D_I) = \dfrac{L(D_I \mid M^+)}{\sum_{D_J D_I} L(D_J \mid M^+)} = \dfrac{L(D_i \mid M^+)}{UB(D_I, M^+)(e^{A_{D_I}}-1)}$ [4]

and sup$(L(D_I, M^+)) = \dfrac{L(D_I \mid M^+)}{\sum_{D_J\ coversof\ M^+} L(D_J \mid M^+)}$ [5]

II.3 A Connectionist Approach of the General Diagnosis Problem Solving [PENG89]

Let x_i be binary variables.$x_i = 1$ if $d_i \in D_I$;$x_i = 0$ otherwise.

Thus to maximize $L(D_I \mid M^+)$ amounts to maximize:

$$Q(X) = \prod_{m_j \in M^+} (1- \prod_{i=1}^{m} (1-c_{ij} x_i)) \prod_{m_j \in M - M^+} \prod_{i=1}^{n} (1-c_{ij} x_i) \prod_{i=1}^{n} \frac{1-x_i(1-p_i)}{1-p_i x_i} \quad [6]$$

This maximization can be get by the use of a two layers neural network.

The units of the first layer represent the manifestations and the units of the second layer represent the disorders.

x_i becomes the activation level of units which represent the disorders. The activation rule of the manifestation nodes is the following:

$$m_j(t)=1-\prod_{i=1}^{n}(1-c_{ij}\,x_i(t))=1-\prod_{d_i\in causes(m_j)}(1-c_{ij}\,x_i(t)) \quad [7]$$

Thus this activation rule is a local computation since it only depends on current activation levels of m_j 's causative disorders which are directly connected to m_j in the causal network.

Since $x_i(0)=p_i$ $\quad m_j(0)=1-\prod_{i=1}^{n}(1-c_{ij}\,p_i)$

The activation rule of x_i is a bit more sophisticated.

Firstly Q(X), which is to maximize is decomposed in $Q(X)=Q'(X-x_i)q_i(x_i(t))$.

Then the activation rule of the node x_i is chosen in order to optimize $q_i(x_i(t))$.

Since $q_i(x_i(t))$ is only function of $x_i(t)$, the use of local optimization for each x_i yields to the optimization of Q(X).

Let $M_i^{+}=M^{+}\cap e(d_i)$ and $M_i^{-}=(M-M^{+})\cap e(d_i)$

$$\text{Let } q_i(t)=\prod_{m_l\in M_i^{+}}\Big(1-\prod_{k=1}^{n}(1-c_{kl}\,x_k(t))\Big)\prod_{m_l\in M_i^{-}}\prod_{k=1}^{n}(1-c_{kl}\,x_k(t))\prod_{k=1}^{n}\frac{1-x_k(t)(1-p_k)}{1-x_k(t)p_k} \quad [8]$$

in which all $x_k(t)$ are considered to be parameters and $x_i(t)$ the only argument of the function $q_i(x_i(t))$.

Note that the first two products in Equation [8] which are local to x_i not over M^{+} and $M-M^{-}$ as in Equation [6].In this sense Equation [8] is a patially localized version of Equation [6] (partially because the parameter $x_k(t)$ for $k\neq i$ are still present.

Viewing $q_i(x_i(t))$ as an objective function and $x_i(t)$ as being constraint to $\{0,1\}$ we decompose the global optimization problem of $D_j(t)$ into local optimization problems of its elements $x_i(t)$:derive whichever of $x_i(t)=1$ or $x_i(t)=0$ will maximize q_i , i.e whichever of $q_i(1)$ or $q_i(0)$ is greater,if all other $x_k(t)$ are fixed.If $q_i(1)>q_i(0)$; $x_i(t)$ should decrease in order to get local optimization.Thus we define the ratio $r_i(t)=\frac{q_i(1)}{q_i(0)}$. It can be proven [PENG89] that:

$$r_i(t)=\prod_{m_l\in M_i^{+}}\Big(1+c_{il}\frac{1-m_l(t)}{m_l(t)-c_{il}(t)x_i(t)}\prod_{m_l\in M_i^{-}}(1-c_{il})\Big(\frac{p_i}{1-p_i}\Big)$$

$r_i(t)$ can rewritten as:

$$r_i(t)=\prod_{m_l\in M_i^{+}}\Big(1+c_{il}\frac{1-m_l(t)}{m_l(t)-c_{il}\,x_i(t)}\Big)K_i \quad [9]$$

The activation rule of the "disorders nodes" can easily be deduced from Equation [9]

Let f(x) be defined as follows:

=1 if x>1
f(x)=-1 if x<-1
=x otherwise

The activation rule for $x_i(t)$ is the following: $\frac{dx_i(t)}{dt}=f\,(r_i(t)-1)(1-x_i(t))$

This differential equation is approximated by the following differences equation: $x_i(t+\Delta)=x_i(t)+f\,(r_i(t)-1)(1-x_i(t))*\Delta$

But if $x_i(t+\Delta)$ is less than 0.0 it is set to 0.0.Thus,as desired $x_i(t)$ is guaranteed to be in [0,1] at any time t.

Experimental studies of this model[PENG89] show that it fits well with its purposes and allows to find out the most probables hypotheses which may explain the obseved manifetations.

III Modelling The Indirect Procedure

III.1 Notations

Let W_i i=1,...p be the possible working modes of the satellite

$H(W_i)=(d_{j1},d_{j2},d_{jk})$ the set of "hiden" disorders in the working mode W_i .(the hiden desorders in a given working mode are the disorders the effects of which are masked in this working mode.For exampme a "solar cell failure " is masked in the working mode "power supplied by battery"

Let $C(m_j)$ be the set of disorders which may be the cause of the manifestation m_j

$M^*(W_i)= \bigcup_{d_j \in D-H(W_i)} M(d_j)$ be the set of manifestations which can be observed in the working mode W_i

III.2 Various Strategies Models

We have studied three possible strategies in the choice of the best working mode in an indirect diagnosis procedure.First we can want to confirm the most likeky explanation of the first phase diagnosis.In this case we have to choose the working mode such that the mean value of this explanation likelihood will be maximum.Thus we have to maximize,with respect to W_j

$E(L(D \mid W_j))= \sum_{M_i^+ \subset M(W_j)} L(D \mid M_i^+)p(M_i^+)$ where M_i^+ stands for all possible set of manifestations and p(M_i^+) stands for the probability of this set of manifestations to be observed with the working mode W_j .Second we can want to eliminate one of the explanations which has been selected in the first phase.For this purpose we have to minimize the mean value of the expected likelihood of this explanation, which amount to minimize $E(L(D \mid W_j))$ .Last, the likelihood of the two most likely explanations may be very close and we can want to maximize the ratio of their mean values of their expected likelihood.In this case we have to look for W_j which maximizes $\dfrac{E(L(D \mid W_j))}{E(L(D' \mid W_j))}$ if D and D' are the two most likely explanations of the first phase.

III.3 Mathematical Approach

In order to achieve these objectives we may use the analytical expression of the relative likelihood and compute it for each possile set of manifestations and make the wheigthed summation of these results for everyworking mode.Because such a way becomes quicklyuntractable when the number of disorders, manifestations and working mode grows, we will show in the next section how the complexity of the computation may be reduced.But before this, we need two easy to compute results:$L(D \mid M^+-\{m_l\})$ and $L(D \mid M^+\cup\{m_l\})$ which stand respectively for the relative likelihood of the hypothesis D when the set of maifestation is respectively M^+ and not m_l and M^+ and m_l A characteristic of satellite failure diagnosis is that we can assume that there is only one failure at a time.Therefore D=$\{d_i\}$ According to Equation [2] we get:

$$L_1(\{d_i\}|M^+= \prod_{m_j \in M^+} (1- \prod_{d_i \in D} (1-c_{ij}))$$

$$L_1(\{d_i\})|M^+)= \prod_{m_j \in M^+} (1-(1-c_{ij}))= \prod_{m_j \in M^+} c_{ij}$$

which yelds to:

$$L_1(\{d_i\})|M^+\cup m_l)= \prod_{m_j \in M^+\cup m_l} c_{ij}$$

$$L_1(\{d_i\})|M^+-m_l)= \prod_{m_j \in M^+-m_l} c_{ij}$$

So:

$$L(\{d_i\})|M^+\cup m_l)=L_1(\{d_i\})c_{il}\alpha_i$$

$$=L(\{d_i\})\}c_{il}\bullet\frac{1-p_i}{p_i(1-c_{il})} \qquad \text{if } m_l\in e(d_i)-M^+$$

$$=L(\{d_i\})c_{il} \qquad \text{otherwise} \quad [11]$$

$$L(\{d_i\})|M^+-m_l)=\frac{L(d_i\mid M^+}{c_{il}} \qquad \text{if } m_l\in M^+$$

$$=L(\{d_i\})|M^+) \qquad \text{if } m_l\in e(d_i)-M^+$$

$$=L(\{d_i\})|M^+)(1-c_{il})\frac{p_i}{1-p_i} \qquad \text{otherwise.} \quad [12]$$

Let $\Delta(m_l,d_i,M^+,x)=x \quad \text{if } m_l\in e(d_i)-M_+$
$=1 \quad \text{otherwise}$

Let $r_{ij}=\dfrac{L(d_i\mid M^+)}{L(d_j\mid M^+}$

$$\frac{L(d_i\mid M^+\cup m_l)}{L(d_j\mid M^+\cup m_l}=r_{ij}\bullet\frac{c_{il}}{c_{jl}}\bullet\Delta(m_l,d_i,M^+,\frac{1-p_i}{p_i(1-c_{il})})\bullet\Delta(m_l,d_j,M^+,\frac{p_j(1-c_{jl})}{1-p_j} \quad [13]$$

$$\frac{L(d_i\mid M^+-m_l)}{L(d_j\mid M^+-m_l)}=r_{ij}\bullet\frac{c_{jl}}{c_{il}} \qquad \text{if } m_l\in M^+$$

$$=r_{ij} \qquad \text{if } m_l\in e(d_i)\cap e(d_j)-M^+$$

$$=r_{ij}(1-c_{il})\frac{p_i}{1-p_i} \qquad \text{if } m_l\in e(d_i)-e(d_j)$$

$$=r_{ij}(1-c_{jl})\frac{p_j}{1-p_j} \qquad \text{if } m_l\in e(d_j)-e(d_i) \quad [14]$$

We also need the a priori probability of a given set of manifestations M_s in a given working mode W_i

$$P(M_s\mid W_i)=\prod_{m_j\in M_s}p(m_j)\prod_{m_k\in M^*(W_i)-M_s}(1-p(m_k)) \quad [15]$$

$p(m_j)=N_j\sum_{d_i\subset H(W_i)}p_i\,c_{ij}$ (remember that $c_{ij}=0$ if $m_j\neg\in M(d_i)$)

N_j is a constant such that $\sum_{M_s\in M^*(W_i)}P(M_s\mid W_i)=1$

From [15] we easily get:

$$P(M_s\cup m_l\mid W_i)=\frac{P(M_s\mid W_i)p(m_l)}{1-p(m_l)} \quad [16]$$

$$P(M_s-m_l\mid W_i)=\frac{P(M_s\mid W_i)(1-p(m_l)}{p(m_l)} \quad [17]$$

We are now able to compute:

$E(L(d_1)\mid M_1^+\mid W_i)=\sum_{M_s\subset M^*(W_i)}L(d_1\mid M_s)P(M_s\mid W_i)$ by computing $L(d_1\mid M_s)$ from $L(d_1\mid M_1^+)$ by succesives use of formulae [2] and [3]

But we have to compute $2^{|M^*(W_i)|+1}$ values and we are going to show in the next section that formulae [11],[12],[13],[14],[16],[17] allow us to minimize the cost of the computation of one value.In the last section we give a theoretical neural ntework which enables us to get the expected mean value of $E(L(d_1) \mid W_i)$ and therefore its maximum or minimum among the avilable W_i

III.4 Complexity analysis and computational cost minimization

If we want to compute $L(d_i \mid M_s)$ we need $1+|e(d_i)|+|M_s|$ operations and $P(M_s \mid W_i)$ needs $|M_s|$ operations.

Thus the computation of one of the term of $E(L(d_1 \mid W_i))$ needs $2|M_s|+|e(d_1)|+1$ operations.Because the mean value of $|M_s|$ is $\frac{|M^*(W_i)|}{2}$ the total computation cost of $E(L(d_1 \mid W_i))$ is $(1+|e(d_1|+|M^*(W_i)|)2^{|M^*(W_i)|+1}$ But,using formulae [11],[12],[13],[14],[16],[17] the computation cost of P($M_s \mid W_i$) is only three operations,the computation of $L(d_i \mid M_s)$ is only one operation and the total cost of computation is reduced to $3*2^{|M^*(W_i)|+1}$.

In order to only use this set of formulae we have to use an algorithm which generates the $2^{|M^*(W_i)|}$ parts of $M^*(W_i)$ in an order such that we can transform each part in the following part only by adding or suppressing an element.This can easily be done by the following recursive algoritm.Let us assume that we want to generate the 2^N parts of a set of N elements $\{a_1,a_2,...,a_N\}$ with respect to the property that two successives parts only differ by one element.Let us assume that we have genrate the 2^{N-1}parts of the subset $\{a_1,a_2,...,a_{N-1}\}$ with respect to the previous property on the order of the parts.Let us assume that the empty set is the first part and that the last part consists of a single element.Let us also assume that the first non empty part is also a single element.Therefore we have $2^{N-1}-1$ non empty parts.In order to get the 2^N parts with respect to the four previously assumed properties we only need to repeat in the reverse order the $2^{N-1}-1$ non empty pars with adding the N^{th} element a_N; in such a way we get $2^{N-1}-1$ parts with a_n begining and ending by a two elements part ;$\{a_i,a_N\}$ and $\{a_j,a_N\}$.This last element can give the part $\{a_N\}$ by suppressing a_j

By concatenation of the two lists of $2^{N-1}-1$ parts and $\{a_N\}$ we get 2^N-1 parts.Therefore with the empty set we have 2^N parts and these parts are ordered with respect to the four previously enouced properties.

Example $\{\emptyset\}\rightarrow|\{a_1\}\rightarrow\{a_1,a_2\}\rightarrow\{a_2\}|\rightarrow|\{a_2,a_3\}\rightarrow\{a_1,a_2,a_3\}\rightarrow\{a_1,a_3\}\rightarrow\{a_3\}|$ $\rightarrow|\{a_3,a_4\}\rightarrow\{a_1,a_3,a_4\}\rightarrow\{a_1,a_2,a_3,a_4\}\rightarrow\{a_2,a_3,a_4\}\rightarrow\{a_2,a_4\}\rightarrow\{a_1,a_2,a_4\}\rightarrow\{a_1,a_4\}\rightarrow a_4\}|$

Remark The expected mean value of the likelihood needs a maximum of computation $2^{|M^*(W_i)|}$ terms because those related to $m_j \in M^*(W_i) \cap M^{+1}$ are already known.The exact number of terms whic have to be computed is $2^{|M^*(W_i)-M_1^+|}$ and the strarting value is in this case L($d_i \mid M_1^+ - M^*(W_i)$)

IV Towards a full connectionist solution

It is obvious that for large $|M^*(W_i)|$ the proposed solution in the previous section becomes intractable.Because the maximum of $L(d_i \mid M_1^+)$ can be found by the means of a connectionnist network the way of a full connectionist solution must be taken into account.The exact computation of a mean value only seems to be done by an Hopfield model network in which each unit represents a part of $M^*(W_i)-M_1^+$ and is linked to the two parts which differ by only one element as it is shown in the previous section.The weight of the link is the factor by which the activity level of a unit must be multiplied in order to get the activity level of the following.But,since this introduces an order for the computation of units activity levels,there is no parallelism in the method.Moreover ,because such a machine with a large number of units is not available nowadays we have not search an algorithm which allows us the use of parllelism,but it must be noticed that the optimum computation cost should be $|M^*(W_i)|-|M_1^+|$ cycles (one for all parts of size 1, one for all parts of size 2 and so on).

Another way is to use a competitive activation model in which the units which compete represent a working mode which is associated with a given disorder the likelihood of which has the required property (i.e to

BEST COPY AVAILABLE

be maximum,or to be the second,or to be minimum in a given set).By similar activation rules (even for minimization for which only the ratio $r_i(t)$ is changed in $\frac{1}{r_i(t)}$) we can determine which working mode has the maximium likelihood with respect to the set of manifestations units.These manifestations units have an activation level equal to the mean value of the binary random variable related to the presence of the manifestation.(See Figure 1)

In this case ,which can easily be implemented on an actual machine with a few thousands of units,we do not compute the expected mean value of the likelihood but the likelihood of the mean values of the manifestations which can be observed during a given working mode.This is different of our initial purpose but can be a good criterion for the selection of the working mode

We have seen that with a very sligthly modification we can define a network which determines the working mode which has the smallest likelihood of the manifestations mean values for a given disorder.Therefore we can help a controller for the choice of the best working mode which would enable him to confirm (respectively eliminate) an explanation.But for the working mode which should maximize $\frac{L(d_i \mid M^*(W_k)\text{-}M_1^+)}{L(d_j \mid M^*(W_k)\text{-}M_1^+)}$ (i.e for which the two explanations d_i and d_j should have likelihoods the most different) we must define another net work.(See Figure 2) This network consists of the both networks previously defined and a set of units which represent each possible working mode.Their activation levels are the ratio between the activation level of the working mode related to an explanation and the activation level of the same working mode related to the other explanation.The unit with the maximum activity level showes the best wor king mode for the choice between the two explanations.

V Conclusion

The framework of a method which allows one to minimizes the number of successives working modes which can be needed for an accurate diagnosis of a satellite failure is established.This method will become tractable when large enough actual neural network become available.Like it can be seen in the previous sections some problems are not yet entirely solved and can only be solved when the characteristics of specific networks will be known.But we also want to outline that this method can be used in a lot of others area;for instance the set size of biological experiments which are needed to type the histocompatibility of cells can be significantly reduced by a stepwise building of experiments plan which is based on the presented method.

REFERENCES

[BOUR86] P.Bourret,P.Ricard,Diagnostic de Pannes Satellites par Systeme Expert
Proc. Journees I.A et Interraction Homme-Machine. Toulouse (France) (Sept 86)

[BOUR87] P.Bourret,A.Cambon,HLA Typing and Interpretation Using Logic Programming
Proc. Artificial Intelligence and Medecine .Marseille (France)(Aug 87)

[PEAR86] J.Pearl,Fusion Propagation and structuring in Belief Networks
Artificial Intelligence 29 pp 241-288 29 (Sept 86)

[PEAR87] J.Pearl ,Distributed Revision of Composite Beliefs
Artificial Intelligence 33 pp173-215 (Oct 87)

[PENG88] Y.Peng,J.Reggia ,Being Confortable with Plausible Diagnostic hypotheses
Information Sciences (Nov 88)

[PENG89] Y.Peng,J.Reggia A Connectionist Model for Diagnostic Problem Solving
IEEE Transaction on System Man and Cybernetic (1989)

[WALD89] J.Wald,M.Farach,M.Tagamets,J.A.Reggia ,Generating Plausible Hypotheses with Self Processing Causal Network
J. of Experimental and Theoretical A.I. (in press)

BEST COPY AVAILABLE

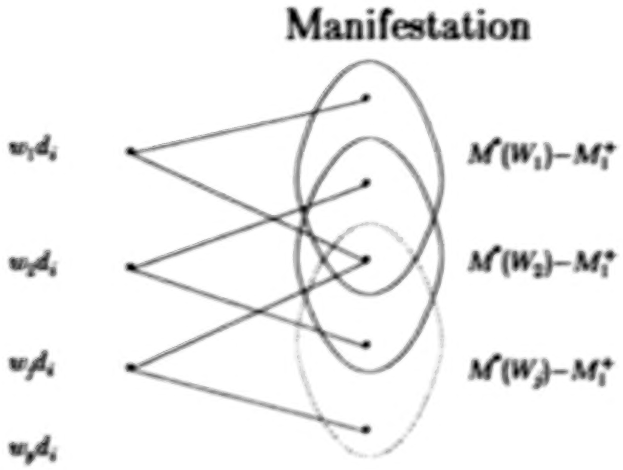

Manifestation
w_1d_i
w_2d_i
w_3d_i
w_pd_i
$M^*(W_1)-M_1^+$
$M^*(W_2)-M_1^+$
$M^*(W_3)-M_1^+$

Figure 1

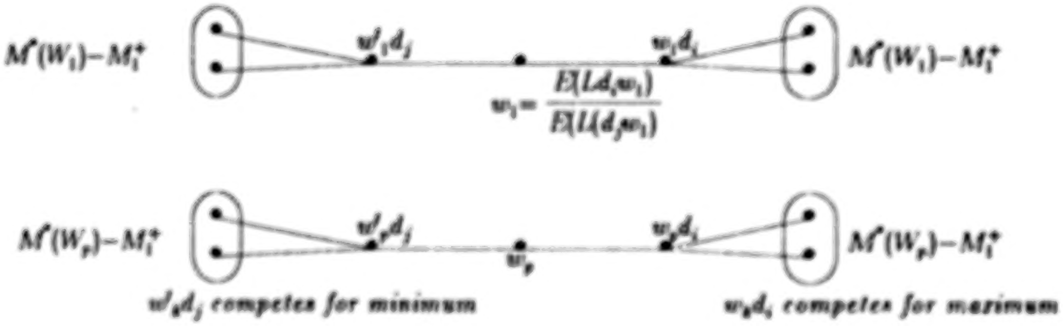

$M^*(W_1)-M_1^+$
w'_1d_j
w_1d_i
$M^*(W_1)-M_1^+$
$w_1=\frac{E(Ld_iw_1)}{E(L(d_jw_1)}$
$M^*(W_p)-M_1^+$
w'_pd_j
w_pd_i
$M^*(W_p)-M_1^+$
w_p
w'_kd_j competes for minimum
w_kd_i competes for maximum

Figure 2

BCAUS PROJECT DESCRIPTION AND CONSIDERATION OF SEPARATION OF DATA AND CONTROL

by

Joy L. Bush and Steven J. Weaver
Computer Sciences Corporation

ABSTRACT

Our experience with the development of a diagnostic expert system has caused us to examine the commonly stated truths that data may be segregated from program control in "generic" expert system shells and that such tools support straightforward knowledge representation. We believe that the ideal of separation of data from program control in expert systems is difficult to realize for a variety of reasons. One approach to achieving this goal is to integrate hybrid collections of specialized shells and tools instead of producing custom systems built with a single "all purpose" expert system tool.

In this report, we try to examine aspects of these issues in the context of a specific diagnostic expert system application, the Backup Control Mode Analysis and Utility System (BCAUS), being developed for the Gamma Ray Observatory (GRO) spacecraft. We present a description of the project, our experience to date, and plans for the on-going system development. A more detailed description of the BCAUS prototype development appears in [1].

1 BCAUS Description

BCAUS is an expert system designed to assist flight operations personnel in diagnosing the cause of a GRO spacecraft autonomous mode transition. The GRO spacecraft has been designed with onboard capability to autonomously *safe* itself, transitioning from a primary operating mode to a backup control (safing) mode, in the event of certain error conditions in the Attitude Control and Determination (ACAD) subsystem. (The logic for the autonomous transitions is described in several papers [2,3].) Flight operations personnel need to understand what error condition triggered the onboard computer (OBC) to order the mode transition and why that error condition occurred so that proper corrective action may be taken. While the actual number of mode transition triggers is small, there are potentially hundreds of underlying causes that could affect these triggers. Thus, the task of diagnosing ACAD failures is nontrivial and involves substantial expertise.

The BCAUS system is intended for use in the GRO Mission Operations Room, where it will be resident on either a 386-PC microcomputer or a Silicon Graphics Iris 4D/20 computer. The initial prototype was developed on a 386-PC-equivalent microcomputer. Input to the system will consist primarily of telemetry from the spacecraft via processing operations within the Multisatellite Operations Control Center, and operator initialization and input. Output from the system will be to the operator only; no output to the spacecraft is involved.

Use of the operational version of BCAUS would typically be initiated when the flight operations team (FOT) realizes, upon achieving contact with GRO during a pass, that the spacecraft is in a backup control mode having made a transition autonomously while out of ground contact. Normally, operations personnel attempt to command the High Gain Antenna and acquire high-speed telemetry and a tape recorder dump. If successful, real-time telemetry is available immediately; the recorded telemetry is available after some processing delay. Once processed, the tape-recorded telemetry is available through the System Test and Operations Language interface, and BCAUS will access these data. It is expected that BCAUS will depend primarily on tape recorder dumps to obtain telemetry data for the diagnostic process rather than using real-time telemetry. This is because GRO has relatively brief periods of ground contact, so it is likely that a mode transition will occur when the spacecraft is not in contact. Also, very little information about the mode transition is preserved in telemetry after the event, so it is necessary to access the recorded telemetry during the time period leading up to and during the mode transition.

The approach taken with BCAUS has been to develop a prototype knowledge base which encodes the expertise spacecraft subsystem engineers use to identify anomalies that trigger autonomous mode transitions. This approach is in contrast to an effort to model the operation and interaction of the spacecraft systems themselves. The focus of BCAUS is on the autonomous mode transitions that result from ACAD safety checks, which is a narrower problem domain than overall spacecraft health and safety. However, it has been necessary to consider the potential effects of failures in many subsystems on spacecraft attitude.

2 Knowledge Acquisition

The knowledge acquisition activities undertaken for the development of the BCAUS prototype involved the following: extensive review of GRO documentation; interviews with Goddard Space Flight Center (GSFC) personnel experienced in spacecraft design, oversight, and operations; interviews with GSFC contractor personnel experienced in spacecraft flight operations and engineering; and interviews with TRW personnel with GRO subsystem engineering, design, and software experience.

The GRO documentation was useful both as an introduction to GRO ACAD operation and as a resource that was revisited as task members accumulated an increased understanding of technical details. However, the most critical aspect in expert systems development is the capture of domain expertise from human experts. The BCAUS prototype is to provide expert assistance in the diagnosis of GRO spacecraft anomalies that lead to an autonomous mode transition, and so requires the input of persons knowledgeable in recognizing anomalous symptoms in GRO flight operations and tracing the causes of those symptoms. However, since GRO has not yet flown, no one yet has that particular expertise. Task members, therefore, sought the input of those persons most closely concerned with the development of the GRO spacecraft and those with flight operations experience on other similar spacecraft. TRW engineering personnel, based in California, possess the most intimate knowledge of the relevant GRO subsystems, and were thus a source of major importance. They were also the least accessible.

Lacking immediate access to the GRO spacecraft subsystem experts, BCAUS developers attempted to acquire sufficient technical knowledge to hypothesize potential failures and symptoms, rather than employing the usual approach of querying the experts for this information. The periods of interaction with the actual experts were then used to attempt to confirm or deny the developers' suppositions. While this has provided sufficient information for a reasonable prototype, more extensive access to experts will be necessary to produce an operational version.

3 Tools Chosen for Implementation

3.1 Prototype Tool Description

KES expert system development software, produced by Software Architecture and Engineering, Inc., was selected for the prototype development. The KES package includes three different expert system shells: a hypothesize-and-test inference engine (HT), a production rule system (PS), and a Bayesian probability inference engine (BAYES). We used the KES HT inference engine, which has the advantage of a built-in diagnostic reasoning technique, *minimal set covering*, which simulates a human-like, sequential, hypothesize-and-test process.

Because the KES HT knowledge base structure was developed for diagnostic applications, the entry of diagnostic knowledge was straightforward, using a cause-manifests-symptoms frame rather than an IF-symptoms-THEN-cause rule. Minimal set covering reasons very efficiently about multiple causes, eliminates impossible hypotheses, and focuses on the most likely hypothesis. The rapid reduction of the search space also generally reduces the number of questions generated by the system for the user to answer. The KES HT implementation also offers a straightforward yet useful handling of confidence factors with reasonable defaults. It uses these confidence factors to either consider or reject hypotheses and to automatically provide a rank ordering of possible solutions from most to least likely.

In BCAUS, a set of symptoms is associated with each possible failure. In a diagnostic problem with a specific set of symptoms present, the KES HT inference mechanism finds all sets of failures that explain or *cover* the set of all symptoms. KES HT reports, as a result of the diagnostic operation, all sets of failures with minimum cardinality (that is, the hypotheses containing the minimum number of failures needed to explain the symptoms). A hypothesis that contains more than that minimum number is not included in the answer. The use of a minimal set cover is based on *parsimony*, the assumption that the simplest explanation is usually the correct one. (See [4] for a full discussion of the theory upon which KES HT reasoning is based.) The KES HT use of parsimony is in keeping with the general philosophy of failure analysis for the GRO project. Based on spacecraft reliability expectations, single-point failures are considered to have a low probability, and multiple-point failures are considered unlikely enough to be largely excluded from the failure analyses. This means that BCAUS will offer a multiple-failure diagnosis only if no single failure is known which can account for the observed symptoms.

In KES HT, diagnostic failure and symptom knowledge is represented in frames. Each frame contains knowledge about a class of failure. Essentially, a KES HT knowledge base description of a problem consists of the name of the problem and a description of the associated symptoms and their symbolic values. Also included is the likelihood of that particular symptom's appearing [i.e., the symbolic certainty factor (SCF)]. KES HT also provides for "setting factors" which are a variation of a symptom. A setting factor is a convenient way of indicating that the absence of a symptom does not eliminate the possibility of a problem, but that its presence makes the problem more likely (the degree of likelihood being indicated by an SCF). Task personnel made extensive use of setting factors to allow a telemetry mnemonic to have a normal (unstated) value in a specific failure case. By only having to explicitly specify non-normal values in the failure frames, failure frame size (and memory usage) was greatly reduced.

3.2 Prototype Implementation

The initial prototype knowledge base entry and debugging took approximately three calendar weeks for two persons. Because of the built-in diagnostic feature of KES HT, there were no rules to enter or debug. The prototype knowledge base contained approximately 100 frames describing possible causes for the eleven triggers.

When KES HT was selected as the software tool for prototype development, some limitations with the tool had been identified. The limitations of KES HT in the area of explanation and justification were more restrictive than anticipated. KES does provide the capability of attaching textual explanations to questions used to elicit user input and to values appearing on the question response menus. This capability was adequate for the purpose of explaining or enlarging on the questions and answers. There is no capability, however, to explain why a question is asked at any given time; the system cannot inform the user that the question's purpose is to confirm or deny a particular hypothesis, for instance. Likewise, while KES HT does provide a trace capability that permits the user to keep track of what hypotheses (potential failures) are under consideration, it does not provide the means to explain how the final conclusion was reached. It should also be noted that KES applications are limited to the 640 kilobyte address space under DOS.

In addition, the possibility exists that the minimal set covering process may eliminate sets of possible failures that would explain the symptoms in certain cases. When the reasoning process develops a number of possible explanations for certain symptoms, each explanation, by the parsimony assumption of minimal set covering, would have the same number of failures, say for example two. A possible explanation of the symptoms in this case that includes three failures would not be considered as a solution. The consideration of *irredundant* solutions by the KES HT algorithm would allow, as a solution, an explanation of the symptoms that included three or even more failures. (Irredundancy says that no set of failures will be included in the solution if the set contains, as a subset, a set of failures that already exist as a solution. See [5] for further explanation of irredundancy.)

3.3 Operational System Implementation

ART-IM, a commercial, off-the-shelf expert system shell from Inference Corporation was selected for the development of the operational BCAUS. ART-IM is a relatively new product; a scaled-down version of the ART system, implemented in C, using the C Language Integrated Production System (CLIPS) as a basis. It is primarily a forward-chaining rule-based system, but provides object-like structures called schemas. A primary reason for this choice was our desire to maintain flexibility in the final choice of delivery platform. ART-IM can be used for development on the 386-PC, and delivered on either the 386-PC or the Silicon Graphics 4D/20.

There are a number of other reasons which contributed to the selection of ART-IM. Although the KES HT frame-based reasoning is very attractive, it carries a number of limitations, including the inability to reason about numeric values without having them transformed to symbolic values, and the lack of full explanation/justification, even with proposed enhancements. ART-IM offers schemas, which can be used in a manner similar to KES HT's frames. ART-IM does not possess the minimal-set-covering type of abductive reasoning that KES HT has, but it does provide rules with very powerful pattern matching capabilities which allows us to tailor operations acting on schemas to our application. The development environment is very good, providing a wide variety of tracing and browsing features.

In evaluating ART-IM as a candidate for BCAUS operational development, we created and ran a small diagnostic system, using schemas which represent disorders with symptoms, in conjunction with rules that acted upon the input symptoms and the schemas. It appeared that ART-IM is capable of implementing the kind of diagnostic system we want.

4 FUTURE DEVELOPMENT

4.1 Porting KES HT Knowledge Base to ART-IM

We are currently converting the KES HT failure frames to ART-IM schemas. We are also extending the previously described small diagnostic system to emulate minimal set covering in ART-IM rules. The emulation of minimal set covering in ART-IM rules has made extensive use of the schema operators provided in ART-IM. The inclusion of procedural control aspects within rules has implications for the development and maintenance of the expert system; this is discussed in Section 5.

4.2 Neural Network Front-end for Telemetry Trending Analysis

The need for having trending data available for the diagnostic task is clear. Many of the failure scenarios known to the knowledge base involve symptoms that require information on the behavior of a value, as opposed to a single reading. For instance, a "noisy" gyro cannot be recognized by one unusual reading. Rather it is the existence of several aberrant, or least inconsistent, readings that is the significant indication. We plan to add a facility to accumulate data over time with regard to specific telemetry mnemonics, and assess the trend of the values. This automated trending analysis program could then assign symbolic trending values, such as "noisy", which would be used as input to the diagnostic reasoning process. In some cases, the operator may be asked to assess the stored values manually or to review the trend determination made by the system. We envision a strip-chart-like display which would be made available for the user so as to allow him to help determine the trend of the data.

We are currently planning to employ a neural network approach to determine the telemetry trends. While it may have been possible to implement trending analysis using ART-IM rules, it was thought that the addition of trend determination rules would overly complicate the BCAUS knowledge base and make it difficult to locate and maintain diagnostic knowledge. In addition, neural networks offer advantages over standard statistical routines in that they can be readily trained to output not only trends, but symbolic determinations of instrument state such as "noisy gyro 1 A channel", in response to time series telemetry inputs. (See [6] for a similar application.) The training data for the neural network trending process will be based on both expert input and empirically derived results. We hope to have simulated data available for use in the latter.

The neural network simulation package we chose is Neural Works Professional II from Neural Ware, Inc. It will be tied into the inferencing part of the system via C function calls from the ART-IM rules. The C functions will either accept symbolic trend outputs from the neural network trending application for all telemetry items, or will run the application for individual or groups of telemetry items.

We intend to gather all needed telemetry items (currently approximately 200 items) for the 10 to 15 minute time period leading up to and through mode transition. The time period is adjustable; determining the appropriate period is expected to be an iterative process. The data archival process will be a pre-processing step offline from the diagnosis. The archived data will be output to a file before it is input to the telemetry trend processor. It is expected that all telemetry items of interest will be archived from the tape recorder playback data. The actual number of samples to process for trending purposes depends on how often the data is expected to change, and whether the telemetry is continuous analog or discrete.

It is possible that multiple neural networks will be used to perform the trending analysis. To improve the performance of the neural network, a network could be trained for each different telemetry item or type of item. It is also possible for the neural network(s) to learn from mistakes made in the assignment of symbolic trends by using the analyst's corrections to re-train or modify the network.

4.3 User Interface

Plans for the user interface include the incorporation of graphic depictions of the relevant spacecraft subsystems in the form of functional block diagrams and causal graphs with potential problem areas highlighted. These will be presented in hierarchical levels allowing movement between top-level subsystem overviews and lower, more detailed, component views.

The user interface has become complicated by the recent shift in project goals away from providing an immediately operational system and toward providing a system which is ready to populate with GRO-specific data, which would then be operational. This latter aim is more complex, because it means that ease of maintenance is even more important than before. As a result, we are considering how to devise easy ways for the GRO FOT to amend the knowledge base, re-train the neural network, and update the graphics interface as needed. Ideally, a tool-kit would be provided to aid the FOT in making the necessary additions and changes; realistically, the funding to do this additional work is not available. Thus, it is of primary importance to consider how modularization of functionality may be achieved throughout system development, and to provide for effective combinations of manual and automated procedures. The separation of the trend analysis and diagnostic functions was motivated by the objective of providing a user-maintainable system.

5 Discussion

The experience of working with tools as different in approach as KES HT and ART-IM for the same application has forced us to realize that isolation of program control from domain knowledge is a central issue in our application. Thus we have had to consider some basic knowledge representation and reasoning issues having to do with the separation of domain and reasoning knowledge, the form with which to best represent these kinds of information, and the advisability of integrating specialized components to accomplish an "expert system" task.

Knowledge-based systems are defined in part by their separation of domain knowledge from program control. Expert systems are a subset of knowledge-based systems that exhibit extensive expertise in a particular domain. The separation of domain knowledge from control of the reasoning process allows the domain knowledge to be more explicit and accessible. If the control and domain knowledge are intermixed, it becomes less clear as to what should be changed to correct or improve the system. It is also less clear as to what the side-effects (if any) of such changes might be. The result is a less flexible and maintainable system.

In simple production rule systems, the domain knowledge is encoded in rules and manipulation of the domain knowledge is handled by the inference engine. This separation is what allows expert system shells to be marketable; developers supposedly have only to code the rules specific to their application, and the shell supplies the general rule reasoning capability. In actual use, however, the separation of domain knowledge and manipulation of that knowledge is not clean. Expert system shell manufacturers find it necessary to provide ways to encode *meta-knowledge*, that is, information, usually procedural, about what to do with the domain knowledge. The shells try to provide the iteration and sequencing control that pure rule-based systems eliminate. However, sequencing rules with such control mechanisms as rule priorities, state variables, and an agenda is more complex and has been likened to programming via side effects [7].

Using KES HT, we built a prototype in which no explicit rules were coded. This is because KES HT supplies a built-in diagnostic inference engine and a frame-based method of representing causes and symptoms. The frames effectively captured the causal rules by implicitly representing the relationship between a failure and possible manifestations. As a result there were no rules to maintain and the knowledge representation was very accessible and independent. Using a shell specializing in diagnosis provided near-complete isolation of program control, which was confined to the inference engine. The domain knowledge alone comprised the knowledge base.

Davis and King [7] discuss applications where the knowledge to be encoded has a strongly sequential character. They state that the inference process can be viewed as a passage through a sequence of states. To control the sequence of rule firings, the system states are tracked with identifiers which are used in the "if" portion of the rule. To fire the appropriate rule, the corresponding state is checked. The resulting rules may look independent as they appear to be individual inference steps, but they are in fact locked together in a tight structure, a sequence of state transitions that defines what to do next. The removal or addition of a rule could destroy the sequence. The point is that some information is inherently sequential. Sometimes we want to know that after we have done W, do X, then do Y, then do Z. The knowledge in such a domain is knowledge of the correct sequence of actions. In such cases the inference engine and knowledge base become nearly indistinguishable.

Part of the dilemma harkens back to early debates in the field about whether expert systems were supposed to encode the "what" or the "how" knowledge. (See [8] for a review of that debate.) If we wish to reason in a particular manner (e.g., minimal set covering), then we must either use an inference engine designed for that purpose or must construct rules (or whatever) to produce the same effect. If we undertake to do the latter, then the problem becomes maintenance of "what" and "how" knowledge separation. One way to simplify this problem is to localize control by breaking the knowledge base into modules; another is to make use of a set of cooperating tools. In the BCAUS project, our approach to knowledge base organization has been to use the ART-IM schemas to encode the "what", or declarative knowledge, and the rules to encode the "how", or procedural knowledge. We have also moved the telemetry trending analysis function to a specialized tool, the neural network. We think that these decisions will help to keep the knowledge base smaller and will facilitate maintenance by restricting the changes the FOT will need to make to the relatively readable schemas. The addition of the neural network actually decreases the control problems of diagnosis, and the strengths of the neural network technology appear very well-suited to the application domain.

6 Conclusions

A number of approaches were used that helped us to achieve some measure of separation between knowledge and control. One approach is to use special-purpose inference engines such as the KES HT minimal set covering diagnostic inference engine. Specialized systems and shells incorporate additional control because they know what kind of control is needed for a particular application. Another approach is to remove the procedural aspects of the problem from the inference engine and employ appropriate algorithms or techniques, using a conventional procedural language or multiple specialized components. Yet another approach is to employ an expert system tool or language that supports procedural constructs within the rules. This is preferable to sequencing rules through the use of state variables.

Our experience on this project has led us to conclude that specialized shells, such as KES HT, may be the best choice for some applications that closely match the shell's design. Specialized shells allow a definite separation between domain knowledge and control, with the latter residing solely in the inference engine. If, however, circumstances force the use of a more generic tool, weaker in terms of built-in application-specific control, the implementor can try to explicitly separate the control aspects from the domain knowledge by making use of the constructs provided by the shell. This is what we have attempted to do in ART-IM by restricting control to the rule set, and encoding domain knowledge in the schemas. Alternatively, or in conjunction with this approach, the system can be broken into a number of cooperating, specialized tools. This permits both further isolation of control, such as use of a C program driver and embedded calls to C programs, and limits the scope of the shell's responsibilities. We are trying to achieve this by using a neural network front-end to perform a specialized trending analysis function, a C program to provided overall control and user interface, and the expert system shell to perform and explain the diagnostic reasoning process.

Approaches such as these that promote the separation of knowledge and control are described in current research and project reports. Rarely acknowledged, however, is that the truism, that knowledge and control are segregated in expert systems, is difficult to obtain. As a result, it is partly responsible for disappointment and misunderstandings about expert systems development and maintenance. The separation of knowledge and control is a goal to aim for rather than a sure foundation upon which to build. Recognition of this fact should help to foster realistic expectations among both developers and users about expert systems and maintain their credibility as a problem solving approach.

ACKNOWLEDGMENTS

The authors wish to thank the project Assistant Technical Representative, Dan Mandl, Code 511.2, Goddard Space Flight Center for his encouragement of this effort. We would also like to thank our supervisor, Doug Carlton, for his assistance and helpful suggestions.

REFERENCES

[1] Computer Sciences Corporation (1988). *Gamma Ray Observatory Backup Control Mode Analysis and Utility System (BCAUS) Prototype Requirements, Design, and Implementation*, CSC/TM/88-6069, prepared for Goddard Space Flight Center.

[2] Jerkovsky, W., L. Keranen, F. Koehler, F. Tung, B. Ward (1986). "GRO Attitude Control and Determination", 86-032, Annual Rocky Mountain Guidance and Control Conference, Keystone, Colorado.

[3] Tai, F., N. Kaskak, K. McLaughlin, B. Ward (1987). "An Innovative Design for Autonomous Backup Attitude Control of the Gamma Ray Observatory", 87-2601, AIAA Guidance Navigation and Control Conference, Monterey, California.

[4] Reggia, J., D. Nau, P. Wang (1983). "Diagnostic Expert Systems Based on a Set Covering Model", International Journal of Man-Machine Studies, 19.

[5] Reggia, J., Y. Peng (1986). "Modeling Diagnostic Reasoning: A Summary of Parsimonious Covering Theory", Proceedings of Computer Applications in Medical Care 1986, Washington, D. C.

[6] Bell, B. and J. L. Eilbert (1988). "Controlling Basins of Attraction in a Neural Network-Based Telemetry Monitor", Fourth Conference on Artificial Intelligence for Space Applications, NASA Conference Publication 3013.

[7] Davis, R., King, J. (1975). "An Overview of Production Systems", Stanford Artificial Intelligence Laboratory, Memo AIM-271.

[8] Winograd, T. (1975). "Frame Representations and the Declarative/Procedural Controversy", 185-210, Representation and Understanding: Studies in Cognitive Science, ed. D. G. Bobrow and A. M. Collins, New York: Academic Press.

Tracking & Data Relay Satellite Fault Isolation & Correction using PACES: Power & Attitude Control Expert System

Carol-Lee Erikson
Westinghouse Electric Corporation

Peggy Hooker
McDonnell Douglas Space Systems Company
TDRS Project, Code 405, (301) 286–8389

Abstract

The Power and Attitude Control Expert System (PACES) is an object oriented and rule based expert system which provides spacecraft engineers with assistance in isolating and correcting problems within the Power and Attitude Control Subsystems of the Tracking and Data Relay Satellites (TDRS).

PACES is designed to act in a consultant role. It will not interface to telemetry data, thus preserving full operator control over spacecraft operations. The spacecraft engineer will input requested information. This information will include telemetry data, action being performed, problem characteristics, spectral characteristics, and judgments of spacecraft functioning.

Questions are answered either by clicking on appropriate responses (for text), or entering numeric values. A context sensitive help facility allows access to additional information when the user has difficulty understanding a question or deciding on an answer.

The major functionality of PACES is to act as a knowledge rich system which includes block diagrams, text, and graphics, linked using hypermedia techniques. This allows easy movement among pieces of the knowledge. Considerable documentation of the spacecraft Power and Attitude Control Subsystems is embedded within PACES.

PACES is being designed and will be delivered on a Macintosh II computer using NEXPERT OBJECT *from Neuron Data as the expert system shell. The graphics oriented user interface to NEXPERT is constructed with AI Vision, a graphics interface tool also from Neuron Data.*

The development phase of TDRSS expert system technology is intended to provide NASA (Code 405) the necessary expertise and capability to define requirements, evaluate proposals and monitor the development progress of a highly competent expert system for NASA's Tracking and Data Relay Satellite Program.

Introduction

The Power and Attitude Control Expert System (PACES) for the Tracking and Data Relay Satellites (TDRS) is intended to assist spacecraft engineers in diagnosing anomalies both on-orbit, and during integration and testing (I&T) of the spacecraft. It is also to be a "knowledge rich" system or information repository, providing engineering explanations of how TDRS Power and Attitude Control Subsystems (ACS) function.

Goals

The major goal of building TDRS PACES is to provide NASA Code 405 personnel with the expertise in actually building a fieldable expert system. This first-hand experience will allow NASA personnel to better judge the expert system components of proposals, and to be more knowledgeable in monitoring and managing expert system projects.

Achieving the primary goal requires actually building a system. Therefore, a prototype (MOORE) was developed, which demonstrated the concept of a diagnostic system. TDRS PACES is currently under development with the goal of being a system which can be implemented. Therefore, it must contain sufficient information to do useful work, and present that information in a usable format.

Objectives

To meet these design goals, TDRS PACES has the following objectives:

- Capture diagnostic knowledge of Power and ACS subsystems, sufficient to allow identification of the most likely component failure at a level which would allow switching to redundant circuitry.
- Maintain information on failed components and anomalies of each spacecraft so that advice can be tailored to the unique characteristics of each satellite.
- Capture block function diagrams, schematics, and illustrations of physical spacecraft components for use by spacecraft engineers.
- Integrate diagrams with each other so engineers can move between the types of representation and levels of detail.
- Integrate the diagrams with TDRS PACES in such a way that the diagnostic function provides initial access to the most appropriate diagram for any identified fault.
- Provide NASA Code 405 personnel with sufficient experience in building expert systems so they can specify their requirements in requests for proposals, evaluate proposals, and judge the quality of expert systems built by contractors.
- Build a system of sufficient complexity and ease-of-use to demonstrate the effective integration of expert system technology in daily orbital operations and test activities. (NASA 88)

Hardware and Software

TDRS PACES is being built using two Apple Macintosh IIs with nineteen-inch color monitors. Each machine has eight megabytes of RAM and a 100 megabyte hard disk. The Macintosh was chosen because of its graphics interface, ease-of-use, and consistent operation across applications. It was also seen as an economical alternative to much costlier workstations. Two machines were used to allow parallel development of portions of the expert system. The large amount of RAM and disk storage were chosen to insure the availability of adequate memory.

The expert system was built using Neuron Data's NEXPERT *OBJECT*. NEXPERT runs on multiple platforms, all of which are general-purpose computers. It includes both rules and an object system – allowing for flexibility in design, and built-in access to external databases. A parallel product from Neuron Data, AIVision, is being used to create a graphical user interface.

A New Image Technology flat bed scanner with associated software, is being used to input some of the graphics to TDRS PACES. Other images are created using several graphic packages, most notably Cricket Draw, Aldus FreeHand, and Adobe Illustrator. Laser writer output of graphics and screen dumps are used with our attitude control expert to eliminate shipping the computer equipment for some of the knowledge acquisition sessions. Shipment of computer resources is necessary because of the remote location of one of our experts.

Situation

Creating an expert system to aid in diagnosing spacecraft anomalies is a difficult task. On the one hand, only a limited number of actions are possible. On the other, it is desirable to know which specific component failed and why. Such knowledge allows corrective action to be taken on future spacecraft, and assessment of risks associated with future performance. A conservative approach must be used when taking action to correct faults as irreparable damage could result if the wrong measure is attempted.

TDRS PACES will operate in the diagnostic process as a consultant. The expert system will not make any decisions for the spacecraft engineers. Rather, it will function as a source of information and advice. TDRS PACES is designed to be used by ground support spacecraft engineers, and by spacecraft engineers during integration and testing of the spacecraft.

Loss of Expertise

There are several characteristics of TDRS diagnostics which suggest the use of expert systems technology. The spacecraft are being built over a prolonged period of time, and they will be used for many years. This means that there will continue to be losses of expertise due to personnel changes. New personnel will need to acquaint themselves with the TDRS satellites.

Each TDRS is designed to have a shelf life of seven years, and a total active life, including storage of eleven years. Since flight 7 is being built now, it is possible that one or more of the satellites will still be used in the year 2000.

At the same time, several key TDRS engineers are nearing retirement. Their expertise will be unavailable in later stages of TDRS operations. Many remaining personnel transfer to other programs as the major work of building the satellites ceases. Without means of capturing engineering expertise specific to the TDRS system, diagnosing faults and suggesting remedies for spacecraft incidents will become increasingly difficult as time passes.

The satellites are being built in an overlapping fashion. At the time Flight 4 was being packaged for shipment to Cape Canaveral, the bus and payload had been mated for Flight 5, while Flight 6 was still in early I&T because many of its modules were used as replacements on Flights 3 to 5.

Only three spacecraft will be on-orbit at any one time: two operational, one as a spare. Other spacecraft will be stored after assembly. In the future, they will be brought out of storage as needed. Currently under construction is Flight 7, a replacement for Flight 2, which was lost with Challenger. Much of the expertise which was available during initial I&T is likely to be unavailable for I&T of Flight 7, or when a satellite is brought out of storage and tested after several years.

Once again, capturing the expertise which is available now will make it easier to deal with gremlins which crept in to a spacecraft while it was being stored. This same expertise will help with the I&T of Flight 7, a task on which there are several engineers unfamiliar with TDRS.

On-orbit Problems

Making repairs to on-orbit spacecraft is a rather limited process. All test data must be collected from telemetry points which were determined during system design. The only repairs which can be affected involve the use of redundant components. Again, all redundancies were part of initial system design.

Because of the limited capacity for on-orbit repairs, satellites are built to be extremely reliable. Thus, relatively few on-orbit problems have been experienced. More difficulties were experienced in the integration and testing phase of the satellites. This I&T data provides some basis for case histories, but the number of faults in I&T is still limited.

Having few problems to comprise an historical record results in limited data from actual cases from which to gather diagnostic expertise. Thus, one must go to the models on which the spacecraft were built to diagnose new faults.

Beyond the Prototype

In 1987-88, a prototype expert system was constructed (Howlin, Weissert, & Krantz 88) to demonstrate the concept of applying expert system technology to diagnosing on-orbit problems in TDRS. The expert system was called MOORE after Mr. Bob Moore, TRW, Redondo Beach, CA, whose expertise was captured in the system. Mr. Moore continues to serve as an expert for TDRS PACES.

The scope of TDRS PACES is wider than MOORE, and it has different objectives. While MOORE was a demonstration prototype, TDRS PACES is designed to be expandable into an implemented system. Different tools are being used, as well as a different approach.

Different Objectives

MOORE was designed from the start to be a demonstration prototype. It illustrated that Artificial Intelligence concepts could be applied to the diagnosis of on-orbit incidents for TDRS. It generated interest in AI technology, and enabled funding of a development system.

TDRS PACES is designed to be a deployed system. Therefore, it must contain sufficient expertise for it to be useful to spacecraft engineers in diagnosing actual on-orbit problems.

Case-based vs Model-based

MOORE was based on reports of on-orbit problems with TDRS Flight 1. Thus, MOORE was at heart a case-based system. In addition to handling exact cases which actually occurred, MOORE was extended slightly to cover problems which were substantially the same as those actually experienced.

There are two major limitations to the case-based approach. First, the number of on-orbit problems is extremely small. Only about 70 anomaly reports were logged over a five year period. These reports covered all aspects of spacecraft operation. Only a small percentage of these were related to ACS difficulties, the only area addressed by MOORE.

The second limitation involves making up problems or playing "what-if..." games. Hypothetical problems could serve as a basis for some expert systems, but not with TDRS. The satellite has many assemblies with thousands of piece-parts each. Hypothesizing could go on for many years, not be inclusive, and produce a system that would be too unwieldy to be implemented. The case approach also yields diagnoses which are much more specific than can be corrected in orbiting spacecraft.

We have chosen a model-based approach for the current expert system. TDRS PACES is based on models of how the power and attitude control subsystems operate, rather than on actual cases. TDRS PACES will provide diagnoses only to the level of replaceable components or redundancies. Then, it will provide a spacecraft engineer access to schematics, function diagrams, and textual information which can aid the engineer in making a more detailed diagnosis of the problem.

Expanded Scope

MOORE was concerned with diagnosing on-orbit Attitude Control Subsystem problems. TDRS PACES expands that scope in several ways. While TDRS PACES will not have the depth of MOORE, it will have considerably more breadth.

TDRS PACES contains information about the Power Subsystem. Mr. Al Gillis, NASA Code 405, GSFC, will serve as the expert for the Power Subsystem. Power would have been a particularly difficult area for using a case-based approach as there have been no power anomalies with Flight 1.

MOORE was designed to serve as a diagnostic system only. In addition to diagnostics, TDRS PACES has an integrated exploration or teaching component. Since many of the spacecraft engineers who could be diagnosing problems with a satellite may have little knowledge of the specific spacecraft, a knowledge intensive component was considered to be essential to an implemented system.

Flight 7 is still to be built and tested, and Flights 5 and 6 must be retested when they are removed from storage. So, TDRS PACES will include information about integration and testing. Including expertise about I&T activities makes TDRS PACES more generally useful.

Interfaces

TDRS PACES uses the interface tools provided by NEXPERT and AIVision for most of the interaction with users. User prompts, graphic interaction, and database accesses are all provided. Where these tools prove inadequate, they are supplemented with routines written in C. The C routines are compiled into the module which makes the NEXPERT library calls. This integration of external routines is an integral feature of NEXPERT.

TDRS PACES will interface with the Reliability Analysis Report database. This interface will allow the engineer to gain access to historical information on the functioning of the spacecraft. The information from the database will be requested by a set of pre-planned queries. The engineer will be presented with options, allowing the selection of pertinent information on a specific spacecraft, or for all spacecraft. The returned data will be reflected to the user through pop-up windows. At this point, the engineer will be able to review the data to determine whether a similarity exists between one of the previous anomalies and the current problem.

The expert system is designed to be used by spacecraft engineers, not software specialists. Therefore, a help facility is being provided which will allow for clarification for questions asked by the system by providing additional explanatory information. In some cases, instructions for moving from one screen to another are provided. Selecting the "WHY" option from the menu in dialog boxes, or clicking on the "HELP" button in graphic displays presents the additional information.

For the majority of interaction with the system, TDRS PACES makes use of the built-in functionality of NEXPERT *OBJECT*. NEXPERT includes a dialog window which prompts the user for answers. AIVision allows for the creation of screens which can display information, and have "hot" areas. These areas can lead to other screens, convey information back to the NEXPERT inference engine, or display the values of objects within NEXPERT. NEXPERT also provides options for allowing text and graphics windows to be opened, displaying static information.

The AIVision screens are used to display graphics, diagrams, and text; providing hypermedia facilities. The dialog window is used to obtain answers to questions. As TDRS PACES grows, some of the dialogs will be replaced by AIVision screens. In other cases, the AIVision screens will contain variable information – information which may be different for each spacecraft.

Diagnostics

Performing diagnostics for the satellites is one of the two main functions of TDRS PACES. The diagnostic system will be built to include both on-orbit and I&T anomalies. All recommendations of the expert system are purely advisory. Spacecraft engineers retain full control over all corrective actions.

Two different types of diagnostics are required by spacecraft engineers in the diagnosis of on-orbit problems. At the action level, a diagnosis is required which will fix the problem – i.e. put or keep the spacecraft on-line, passing messages back and forth between other spacecraft and the White Sands Ground Terminal. In addition, the engineers must determine the exact component which failed. Individual components are not replaceable.

Identifying the failed part is a "deep" problem compared to the "shallow" problem of finding the failed assembly or subassembly of which the failed component is a part.

The shallow knowledge represents a much smaller body of knowledge than the deep knowledge. It is possible to identify and represent this shallow knowledge in a reasonable length of time.

System Structure

Mr. Moore, the attitude control expert begins the diagnostic process by asking a standard set of questions. This set of questions is used as the introduction to the diagnostic phase of TDRS PACES. The attitude control questions are supplemented with questions about the time of year to gain initial information about the Power Subsystem. Because the spacecraft is eclipsed by the earth for short periods of each day in the fall and spring, different operations related to charging and using the batteries are performed before and during these eclipse periods.

The initial questions for on-orbit satellites, along with the selectable responses in brackets, include:

- Which spacecraft is experiencing problems? [F1, F3, F4]
- On what day did the error occur (DD-MM-YY)?
- At what time did the error occur (HH:MM:SS)?
- Is this a valid spacecraft event? [Yes, No]
- Is the spacecraft in fail-safe mode? [Yes, No]
- In what mode of operation was the spacecraft when the problem was experienced? [Earth, Inertial, Normal, Sun]
- What function were you performing when the problem was first noticed? [Antenna Slew, Momentum Dump, Monitoring, ...]

Figure 1 illustrates the three types of dialog window provided by NEXPERT. They allow entry of boolean values, selection from a list of choices, and entry of text or numeric data. The default screens are used in all cases where they are not superceded by a specific graphic screen.

Diagnostic fault trees are constructed for each redundancy. The trees specify path(s) leading to the diagnosis of a problem with the (sub)assembly. When a (sub)assembly is found to be at fault, the recommended action includes switching to the redundant (sub)assembly. The fault trees are implemented as production rules.

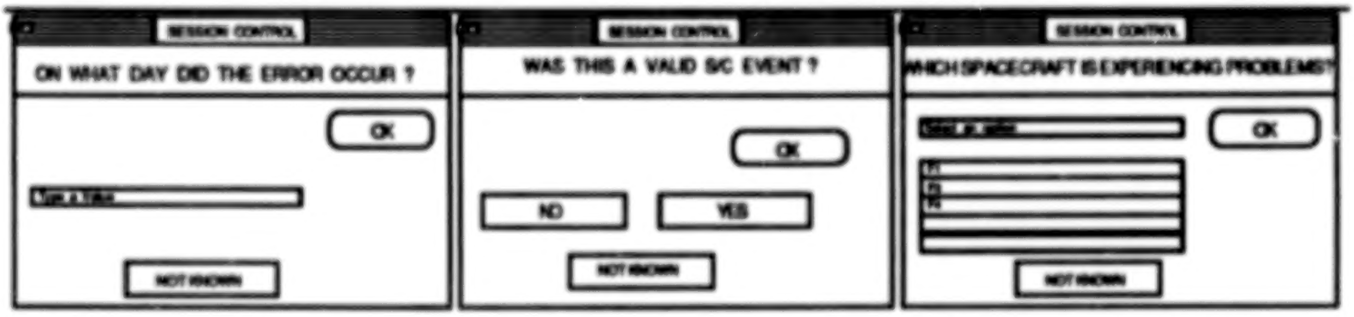

SESSION CONTROL
ON WHAT DAY DID THE ERROR OCCUR ?
OK
NOT KNOWN
SESSION CONTROL
WAS THIS A VALID S/C EVENT ?
OK
NO
YES
NOT KNOWN
SESSION CONTROL
WHICH SPACECRAFT IS EXPERIENCING PROBLEMS?
OK
F1
F3
F4
NOT KNOWN

Figure 1. Default response screens. *There are three default response screens provided by NEXPERT. The one on the left allows entry of numeric values from the keyboard. The middle one allows for response to "Yes/No" questions. For text strings, NEXPERT presents all possibilities in a scrollable list as in the box at right. In all cases, an option is provided for the user to indicate "Not Known" – a response which can be used to bring up other questions.*

BEST COPY AVAILABLE

Assemblies and subassemblies are considered to be objects in TDRS PACES. Separate objects are maintained for each satellite to allow recording failed components in specific spacecraft. The level of modeling extends to where components have redundancies. The objects are grouped into classes.

Data which form the objects are kept in external database files for easy updating. When TDRS PACES is run, objects are created from the latest data.

Exploration

Perhaps the most important feature of TDRS PACES will prove to be the exploration facility. While the exploration function contains no complex reasoning, it provides the type of information which a spacecraft engineer will need in order to understand the Power and Attitude Control Subsystems of TDRS. The exploration facility is designed to be relatively large and extensible.

The exploration facility contains schematics, function diagrams, textual descriptions, and illustrations of spacecraft components. They are linked by "hot spots" - areas on the screen which when clicked on, bring up other screens.

Two types of linkages are available. Within a type of screen, similar screens with greater or lesser detail can be accessed. A screen of one type also allows access to other types of screens – other presentations of the same material.

Figure 2 contains sample screens, illustrating their linkages. A diagram of the attitude control sensors and actuators will connect to more detailed diagrams of the sensors and actuators. By clicking the "hot spot" on a reaction wheel assembly in the sensors and actuators diagram, a graphic of the reaction wheels is presented. Then, clicking on the Function button brings up a text description of the reaction wheels. From the graphic of sensors and actuators, one can gain direct access to the function block diagram or an overall text description of the ACS.

The information contained in the Exploration facility comes from briefings and project documents. The briefings were given by Mr. Bob Moore and Al Gillis over several years and represent information they deem to be important. This briefing material is supplemented by information obtained from Messrs. Moore and Gillis in knowledge acquisition sessions. The primary document used for obtaining graphics and explanatory information is *The TDRSS Spacecraft Systems Manual* (TRW 85). Graphics were entered through drawing programs or by using an image scanner.

Conclusions

Through the process of building TDRS PACES, Code 405 is gaining considerable experience with the practical aspects of developing, managing, and monitoring an expert system project. This experience will prove to be of considerable value as more projects contain requirements for including expert systems.

In many respects, TDRS PACES reflects the experience of many real-world systems regarding actual artificial intelligence content. The most useful feature of the system is the knowledge-rich, hypermedia section. While this contains much information and expertise, that expertise is captured in the graphic interface, not in the inferencing process. However, it is important to remember that TDRS PACES is designed to do useful work – work which saves time and money – not to break new ground in theoretical areas.

References

Howlin, K., Weissert, J., & Krantz, K. "MOORE: A Prototype Expert System for Diagnosing Spacecraft Problems" *1988 Goddard Conference on Space Applications of Artificial Intelligence. NASA Conference Publication* 3009, Greenbelt, Maryland; 1988.

NASA Code 405. *PACES System Requirements Document (405-F7-ES-001).* Greenbelt, Maryland; 1988.

TRW, Defense Systems Group. *TDRSS Spacecraft Systems Manual (TM 02-54).* Redondo Beach, California; 1985.

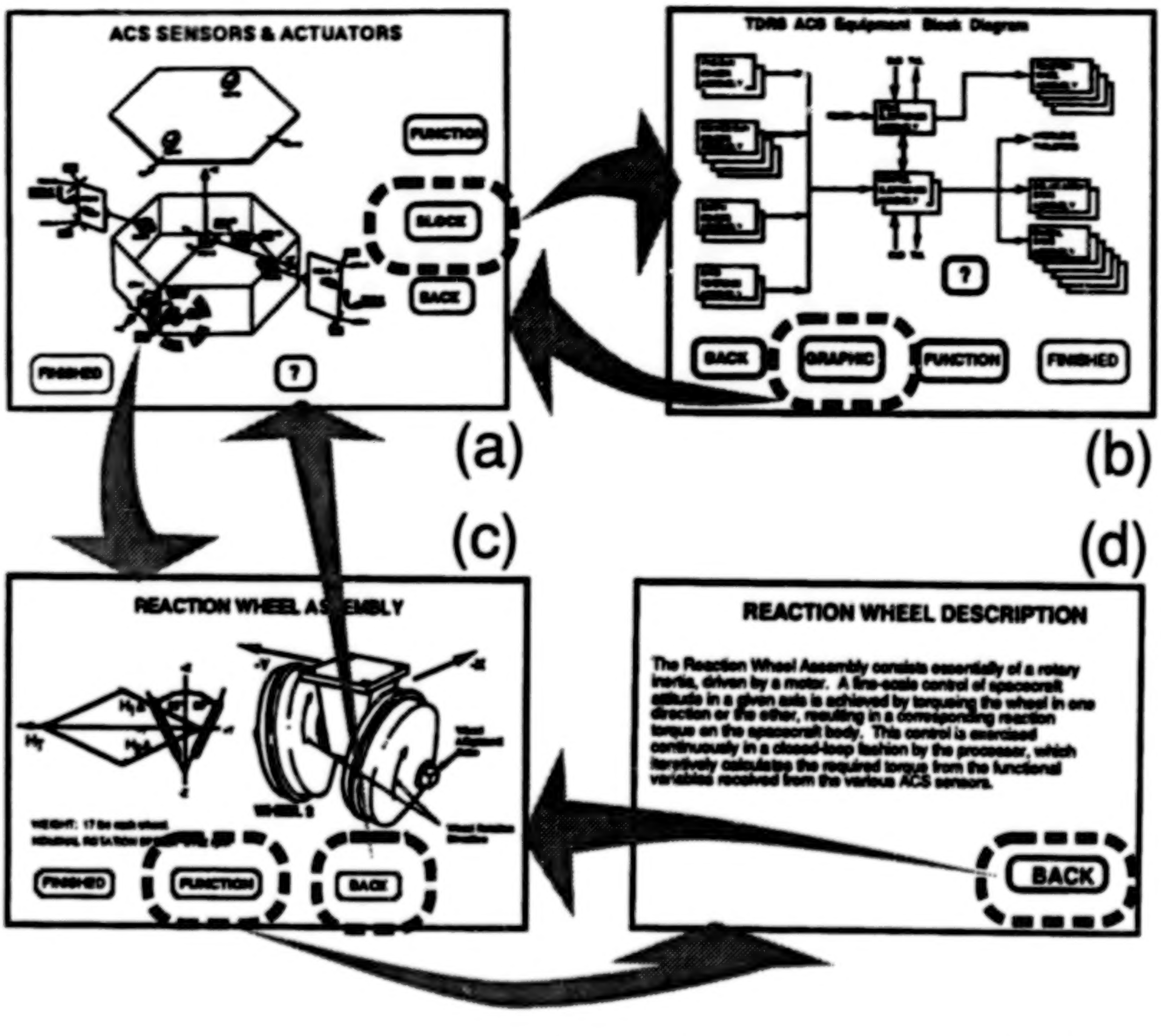
Acknowledgments

The authors would like to thank the following indispensable contributors:

Mr. Laurence Goodman, Project Support Manager, who conceived and manages the project and its development.

Mr. Robert J. Moore who serves as our Attitude Control Subsystem expert, and suggested the presentation format for PACES.

Mr. Al Gillis who serves as our Power Subsystem expert.


ACS SENSORS & ACTUATORS
FUNCTION
BLOCK
BACK
FINISHED
?
TDRS ACS Equipment Block Diagram
?
BACK
GRAPHIC
FUNCTION
FINISHED
(a)
(b)
(c)
(d)
REACTION WHEEL ASSEMBLY
FINISHED
FUNCTION
BACK
REACTION WHEEL DESCRIPTION
BACK

The Reaction Wheel Assembly consists essentially of a rotary inertia, driven by a motor. A fine-scale control of spacecraft attitude in a given axis is achieved by torquing the wheel in one direction or the other, resulting in a corresponding reaction torque on the spacecraft body. This control is exercised continuously in a closed-loop fashion by the processor, which iteratively calculates the required torque from the functional variables received from the various ACS sensors.

Figure 2. Types of exploration screens and their linkages. *From a graphic of the ACS sensors and actuators (a), clicking on the "Block" button brings us a block diagram of the ACS system (b). Clicking on the "Graphic" button on the block diagram, brings up the graphic of the system. Clicking on the reaction wheel section of the ACS Graphic (a), brings up a detailed graphic of the reaction wheels (c). Selecting the "Function" button brings up a textual description of the reaction wheels (d).*

BEST COPY AVAILABLE

Spacelab Life Sciences-1 Electrical Diagnostic Expert System*

C. Y. Kao, W. S. Morris

General Electric Government Services
Mission Integration Office
1050 Bay Area Blvd.
Houston, Texas 77058 (713)488-9005

ABSTRACT

The Spacelab Life Sciences-1 (SLS-1) Electrical Diagnostic (SLED) expert system is a continuous, real time knowledge-based system to monitor and diagnose electrical system problems in the Spacelab. After fault isolation, the SLED system provides corrective procedures and advice to the ground-based console operator.

The SLED system uses the Unit (frame) of KEE to represent the knowledge about the electrical components and uses KEE-Bitmaps to represent the electrical schematics. The diagnostic logic, stored as a set of LISP structure, mirrors that in the malfunction procedures defined in the Spacelab Flight Data File Malfunction Procedures Handbook (JSC-18927) of NASA. The SLED system utilizes downlink telemetry data as input. The system performs some initial screening of the data in order to recognize patterns representing serious problems, and updates its knowledge about the status of Spacelab every 3 seconds. Important parameters are monitored via Active-Values within KEE. The Active-Value kicks off the diagnostic analysis to determine the source of the problems if any problem has been identified. The system supports multiprocessing of malfunctions and allows multiple failures to be handled simultaneously. The user can examine each of the reported problems and receive corrective advice related to each problem. Information which is readily available via a mouse click includes: general information about the system and each component, the electrical schematics, the recovery procedures of each malfunction, and an explanation of the diagnosis.

A rich set of user interfaces is provided in SLED. Various tools have been included which allow a non-programmer to define new diagnostic procedures, define and update schematics of the electrical system, and to change the SLED model by changing the graphical representation. Each tool and function has explanatory prompts to aid the user.

*** This work is performed by GE Government Services, Johnson Space Center, Houston Texas in support of the NASA Mission Management Office, Mission Manager D. Womack. Government Contract NAS9-17884**

1. INTRODUCTION

The Spacelab Life Sciences-1 (SLS-1) Experiment Electrical Diagnostic (*SLED*) system is a continuous, real time knowledge-based system to monitor and diagnose experiment equipment electrical system problems in Spacelab. After fault isolation, the *SLED* system provides corrective procedures and advice to the ground-based console operator. Operation of the system is to be continuous throughout the SLS-1 mission. The inputs to *SLED* are a stream of parameters downlinked from the Tracking and Data Relay Satellite (TDRS) system. This paper covers the functionality of the *SLED* system , and compares it with the current conventional approach for diagnosing experiment electrical problems in Spacelab.

Subsequent sections to follow will include a background description on how the *SLED* system evolved, the application domain characteristics and issues, the system design overview, the current status, and the conclusion comments.

Background

The console operation of the Payload Operations Control Center (POCC) is an important but tedious task. Some of the Avionics Systems Payload Systems Engineer (PSE)'s tasks involve trouble shooting and determining the status of hardware between the Spacelab/Experiment/Mission Peculiar Equipment (MPE) Systems. They also assist in resolution of hardware problems with Johnson Space Center (JSC) Mission Control Center (MCC), Principal Investigator (PI) teams, Science Monitoring Area (SMA), and POCC Cadre positions.

The current approach for the detection and resolution of Spacelab experiment electrical failures begins with the PSE observing console displayed data for out-of-limit errors. Once an error is detected, the PSE determines what type of error has occurred and searches through the Flight Data File Spacelab Malfunction Procedures, [JSC-18927], to find, isolate, and provide recovery procedures for the electrical failure. The procedure that the PSE follows includes looking at other consol displayed parameters to determine the status of interrelated hardware. The PSE then looks at the schematic for the system, which is found in another handbook, S/L Systems Handbook, [JSC-12777C]. Also, the procedure may require communicating with the crew to perform some type of activity in the Spacelab Module, which may be necessary to diagnose the failure correctly. Having isolated the failure, the malfunction procedure includes or identifies a procedure (Sub System Recovery (SSR)) for recovery to a nominal configuration. These recovery procedures list what actions are to be taken by the crew, the equipment that has been lost due to the failure, crew indications, and additional notes. The SSR procedures may include communicating with the crew/POCC to perform some type of action in order to resolve the failure. The task is very time consuming and can produce many human errors. This is the flow that the crewmember/PSE has to perform for every malfunction that occurs. If multiple errors occur at one time this not only adds to the confusion for the PSE, but also introduces a probability of more human errors.

The *SLED* system was devised from a General Diagnostic system, which was initiated in late 1986, to aid the tedious task of diagnosing Spacelab failures. This system's main purpose is to make the console operation task of the

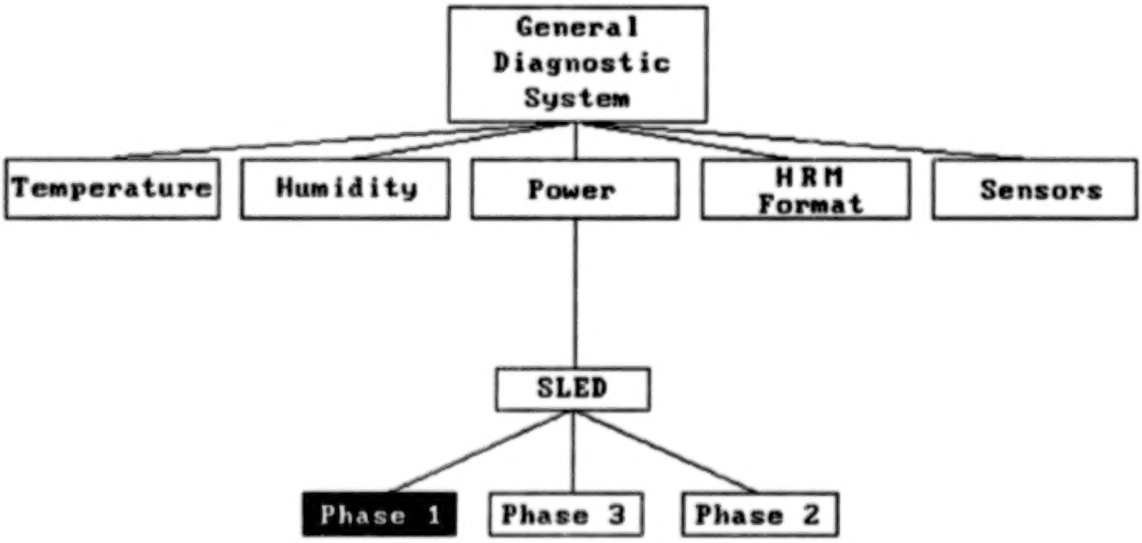


Figure 1. How the current SLED system originated.

PSE easier. Its capabilities were to include diagnostics concerning power, temperature, the high rate multiplexer (HRM) format, humidity, and sensors (Figure 1). Each of these areas constitute comprehensive malfunction recovery procedures. In other words, the intended expert system will be able to diagnose power problems, high and low temperature problems, HRM format irregularities, problems related to the high and low humidity and problems resulting from bad sensors. Designing an expert system having all these diagnostics to work in a real-time useful manner is a very large task. After spending more then half a year in prototyping, the initiators decided to break the large problem into smaller problems. This is how the *SLED* system evolved, from the General Diagnostic system. As with each of the other subdomains, temperature, humidity, HRM format, and sensors, can be their own expert system. Integrated together as a whole, they can interact as one large domain expert system. The *SLED* system was initially initiated to include the entire electrical system from the electrical power distribution system (EPDS) up to and including the experiment hardware. This system was still a large problem. The problem was simplified by splitting it into three phases. Phase one implements the EPDS, phase two covers the hardware beginning where the EPDS left off up to the "box" that contains the experiment hardware, and finally, the third phase covers the experiment hardware. Phase one is completed and is described in detail in this paper.

Objective of SLED

The objective of the *SLED* system is to develop a continuous, real-time expert system to monitor and diagnose the electrical power problems in the Spacelab and the experiment equipment. The primary goal is to help the PSE to monitor the telemetry data and to diagnose the malfunction by firing the diagnostic procedure automatically when an out-of-limit condition is detected, especially in the event of multiple-failures when the PSE is most confused by the abnormal telemetry data. The secondary purpose of the *SLED* system is to aid the PSE in fixing these problems by calling up the malfunction procedure and the SSR procedure. Because we have covered all of the malfunctions that appear in the EPDS schematics, we anticipate that the *SLED* system will find the causes of most of the system faults. In the other cases when the *SLED* system failed to identify the problem or failed to give correct malfunction procedures, the system should jog the PSE's memory enough to solve the problem with the help of *SLED*'s schematic display facility and the available status information of every component in the schematics.

2. APPLICATION DOMAIN and ISSUES

The basic Spacelab EPDS distributes DC-Main, DC-Essential and 400 Hz AC power to Spacelab experiment equipment and also provides the necessary power to Spacelab subsystems. The EPDS receives its DC power (+28V nominal) from Orbiter hydrogen/oxygen fuel cells through the Orbiter bus system which, in turn, is connected to the Spacelab Power Control Box (PCB) and the Spacelab Emergency Box. The AC power is generated from the DC main power by the Spacelab 400 Hz inverters. Dedicated Spacelab Subsystem (S/S) and experiment DC/AC inverters, which receive their primary power from the main DC power bus via the PCB, supply three-phase AC power to the S/S equipment and experiment equipment. For experiment caution and warning sensors, safing command actuators and critical experiment equipment needing power during all mission phases, a dedicated experiment essential power bus is routed through the Spacelab Module. This power bus receives its essential power from the Spacelab Emergency Box, which, in turn, is powered by the Orbiter auxiliary DC power busses.

Using Expert System technology in Space Mission Operations is a focus of current research in recent years since the promising benefits of the technology and the strong desire of automating the mission planning and mission control tasks to keep the Aerospace exploration cost down. [Dickey & Toussaint, 1984] describe a prototype expert system for the integration of housekeeping subsystems on board a manned space station. [Silverman, 1986] talks about a distributed expert system testbed for Spacecraft ground facilities. [Rook & Odubiyi, 1986] describes a prototype expert system to provide real-time aid/advice for ground based satellite orbit control operations. [Muratore, Madison & etc. 1988] describes the development of the real-time expert system prototype for shuttle Mission Control Center as a five layer model to integrate various monitoring and analysis technologies such as digital filtering, fault detection algorithms, and expert systems. [Hamilton, 1986] describes a prototype expert system application to detect and diagnose faults in attitude control systems. Besides these listed, there is an annual conference sponsored by NASA, Goddard Conference on Space Applications of Artificial Intelligence, which focuses on Artificial Intelligence in Space. Several papers presented in this conference are relevant to our target application domain of developing a monitoring and diagnostic expert system for Spacelab Power Subsystems that is capable of supporting real-time operation and diagnosing multiple faults. In particular, [Wilkinson, Happell & etc., 1988] described a prototype fault isolation expert system for TDRSS application in a real-time on-line environment, which is similar in nature to our system.

Three special characteristics of the application domain is important to the design and development of the expert system in space application.

1. **The domain is not mature and there is not a so called "expert" in the domain.** This implies that the system design should be flexible and the designer/implementer should be willing to change the system architecture if needed.

2. **The resulting expert system should be a continuous, real-time system, which is admitted to be difficult to achieve within current expert system technology.** As [D'Ambrusio, Fehling & etc., 1987] pointed out, few knowledge base systems have been developed for real-time applications. How to meet the real-time requirements in an expert system is an important issue. [Griesmer, Hong & etc., 1984] reported the implementation of the YES/MVS, which is a continuous real-time expert system, they also outlined the real- time requirements for expert systems. Specifically, they discuss how to modify OPS5 ([Forgy, 1981]) in support for real-time tasks namely: 1. Speed considerations; 2. Initialize an action like production firing at a given time; 3. Fast communications between modules; 4. Need for explicit control (by the operator); and 5. Some specific requirements of continuous operations. These continuous operation requirements includes: (a). Inference engine should not terminate; (b). Automatic restart capability; (c). Remove "garbage" work-memory element. Similar requirements for expert systems in space applications has been outlined in [Leinweber, 1987] namely: 1. High-Speed context-sensitive rule activation; 2. Efficient recycling of no-longer-needed memory elements; 3. Interactively accept command sequences from operators; 4. Fast communications between multiple expert systems. These real-time requirements affect our design decision. [Wilkinson, Happell, & etc., 1988] discuss the "hard" real-time requirements, which is "provide outputs by some deadline time". They also pointed out that in an AI system, the time taken to complete a calculation is nondeterministic . The state-of-the-art for such problems are often to build something to see if it will work in real time. Performance problems are solved via the bigger hammer theory: if it is not fast enough, buy a faster machine. They call this a "soft" real-time system. We are in the same boat, we plan to deliver the SLED system as a "soft" real-time system. We will discuss these issues later.

3. **This system should be able to detect and diagnose multiple faults, which is a tough problem too.** As we all know, fault diagnostic systems is one of the traditional expert system application areas,[Hayes-Roth, Waterman & Lenat, 1983]. For example, MYCIN ([Buchanan & Shortliffe, 1984]) is a classical example of a medical diagnostic system.

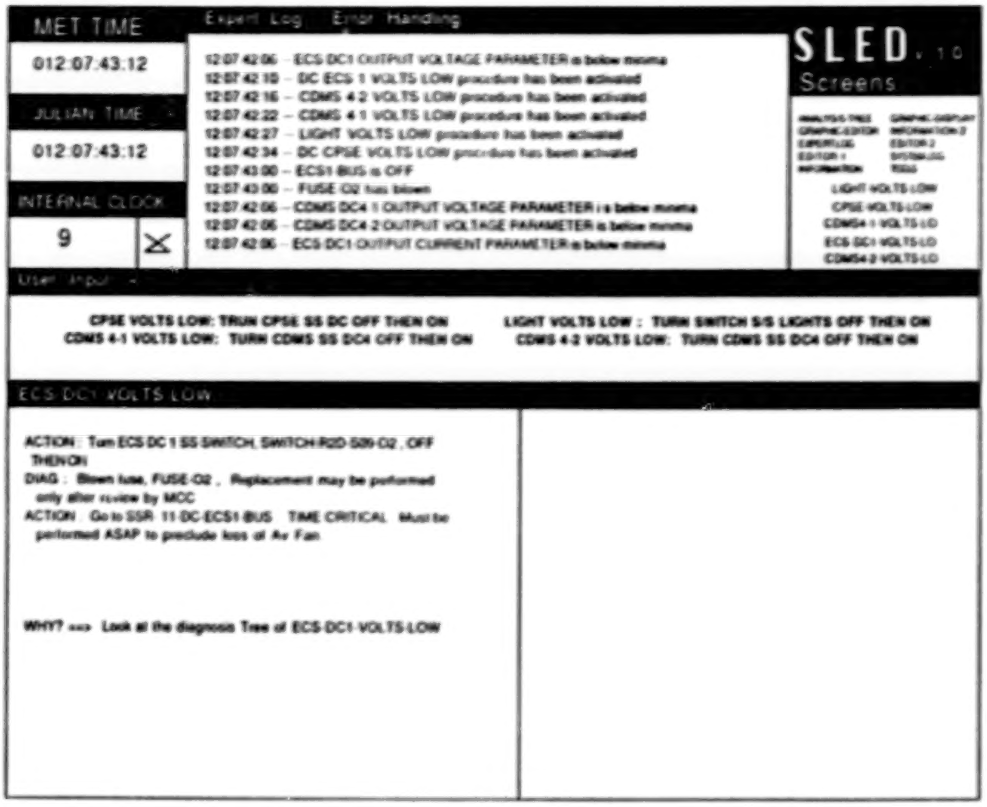
Instead of using shallow knowledge in electrical trouble shooting, it is argued that a design-model-based approach, rather than the traditional empirical-rule-based approach is necessary for electronic diagnostic systems. In this approach, the only available information is the system description, i.e. its design and structure, together with some observations of the system's behavior and a statistical characterization for the type of failure. [Davis, 1983] argued about using the first principle and structure knowledge in electronic trouble-shooting. [Milne, 1987] designed a system using "second principles", which are rules that an electronic engineer would use when diagnosing a circuit during a trouble-shooting task. [Cantone, Pipitone & etc., 1983] proposed the model-based probabilistic reasoning for electronic trouble-shooting. [Pipitone, 1984] used the qualitative causal model and generic component knowledge to produce a model for diagnostic reasoning. [Taie & Srihari, 1986] proposed a hierarchical device representation scheme using instantiation rules and structural template in a semantic network with procedural attachment for function description of a device. [Maletz, 1985] used context graphs combined with electrical system structures, diagnostic tests, and symptom knowledge to support multiple fault hypothesis and reasoning about the abilities of testing to discriminate among a collection of possible faults. [Geffner & Pearl, 1987] used belief networks to represent causal knowledge of system behavior, and using Bayesian inference for doing diagnosis. The distributed scheme then uses the independices embedded in a system to decompose the task of diagnosing the overall system into smaller sub-tasks of diagnosis for subparts of the net, then combined them together. Geffner & Pearl claim that the decomposition yields a globally-optimum diagnosis by local and concurrent computation using message- passing algorithms and attaining linear time in single-connected networks.

3. SYSTEM DESIGN OVERVIEW

The *SLED* system has been developed using the Texas Instruments (TI) Explorer workstation and the Knowledge Engineering Environment (KEE) expert system building tool ([KEE 2.1]). The *SLED* system uses the Unit (frame) of KEE to represent the knowledge about the electrical components and uses KEE-Bitmaps to represent the electrical schematics. When the downlink data is received, the values are put in their appropriate Unit within KEE. Utilizing the Active-Values of KEE, malfunction recovery procedures are activated. The decision for the activation of the recovery procedure is based upon a maximum and minimum value for that parameter.

SLED Approach

The *SLED* system approach for the detection and resolution of Spacelab experiment electrical failures incorporate system monitoring of the downlinked parameters for out-of-limit conditions. The software is designed to standby when no abnormal event occurs. The user will have no interaction with the system besides watching the telemetry data displays if he likes. When the system receives an out-of-limit condition a malfunction recovery procedure is activated. Once the procedure is activated a unique window is created for that malfunction. The diagnostic procedure may obtain any other value it needs in order to diagnose the out-of-limit condition. It may also request crew actions to be performed before the diagnosis can continue. The action that the crew is to take is displayed in the User Input window. Once the action has been completed the user would click on the displayed action and the recovery procedure resumes execution. When the system wants to let the user know something about the diagnosis, it displays this information in the unique window that was created upon activation.



```
12:07:42:06 -- ECS DC1 OUTPUT VOLTAGE PARAMETER is below minima
12:07:42:10 -- DC ECS 1 VOLTS LOW procedure has been activated
12:07:42:16 -- CDMS 4.2 VOLTS LOW procedure has been activated
12:07:42:22 -- CDMS 4.1 VOLTS LOW procedure has been activated
12:07:42:27 -- LIGHT VOLTS LOW procedure has been activated
12:07:42:34 -- DC CPSE VOLTS LOW procedure has been activated
12:07:43:00 -- ECS1 BUS is OFF
12:07:43:00 -- FUSE O2 has blown
12:07:42:06 -- CDMS DC4 1 OUTPUT VOLTAGE PARAMETER i s below minima
12:07:42:06 -- CDMS DC4 2 OUTPUT VOLTAGE PARAMETER is below minima
12:07:42:06 -- ECS DC1 OUTPUT CURRENT PARAMETER is below minima
```

CPSE VOLTS LOW: TRUN CPSE SS DC OFF THEN ON
CDMS 4-1 VOLTS LOW: TURN CDMS SS DC4 OFF THEN ON
LIGHT VOLTS LOW : TURN SWITCH S/S LIGHTS OFF THEN ON
CDMS 4-2 VOLTS LOW: TURN CDMS SS DC4 OFF THEN ON

ACTION : Turn ECS DC 1 SS SWITCH, SWITCH R2D-S20-O2 , OFF THEN ON
DIAG : Blown fuse, FUSE-O2 , Replacement may be performed only after review by MCC
ACTION : Go to SSR 11-DC-ECS1-BUS TIME CRITICAL Must be performed ASAP to preclude loss of Av Fan

WHY? ==> Look at the diagnosis Tree of ECS-DC1-VOLTS-LOW

Figure 2. Malfunctions Window Display

Some of the text may be mousable so that the user may select it. Once the system has finished its diagnoses, the user will be notified. The mousable text that may be included in the text displayed by the system will allow the user to obtain additional information uniquely related to the out-of-limit condition being diagnosed. The type of additional information includes schematics of the related electrical components, specific information on an electrical component, SSR procedures, and a tree display showing how the system obtained its diagnosis. With this approach the user does not have to search through any manuals. All information is readily available. The system can detect all out-of-limit conditions that are pre-defined in the system. It is also capable of handling more than one out-of-limit condition, and can diagnose them simultaneously. (Figure 2)

Meet Real-Time Requirements

For supporting continuous operations, the main process of the *SLED* system will run forever. To protect it from aborting, we bounded the "Expert Log" window to be the terminal I/O and query I/O window while the *SLED* window frame is initially displayed. These bindings cause the Lisp machine to use the "Expert Log" window in error handling. If a system bug was ever to occur, and the user does not want to handle it, he can simply press the Abort-key to re-start the *SLED* system. We also put re-start capability in the process for data acquisition. Therefore, if any data connection problems occur, the two synchronized processes in (the simulation of) the Data Acquisition Layer will attempt to re-connect the data link, which keeps the data acquisition process running continuously.

One drawback in the KEE expert system shell is that the rule system is very slow. In order to meet the high-speed requirement of a real-time system we avoided using the rule system of KEE for the fault diagnostic process. Instead, we defined a frame-type Lisp structure to explicitly represent the diagnostic procedures. We also included a Lisp function to execute the diagnostic procedure defined in the frame. This way, the diagnostic reasoning is just a data-driven traversal of the malfunction specific analysis-tree. The execution of the diagnostic logic is tremendously fast. In fact, the time taken by the malfunction window, that is being displayed, and the user input is the only significant time consuming portion in the fault-diagnostic process. Of course, the decision of explicitly representing the diagnostic procedure may have some drawbacks. But the simplicity of the structure of each malfunction diagnostic procedure described in the Spacelab Malfunction Procedure Handbook makes this approach feasible. If the performance of the rule-system within KEE does not dramatically improve, and we are faced with more complicated diagnostic procedures, in order to meet the fast-speed real-time require-

BEST COPY AVAILABLE

ments, we may need to use other rule- engines, e.g. OPS5 ([Forgy, 1981]).

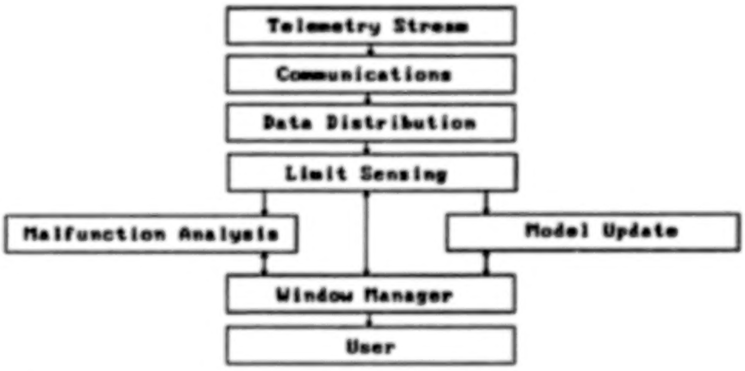
The architecture of Lisp-machines and the availability of some system software tools, in particular the stack group, the scheduler, the process structure, and the signal mechanism, make it easy to handle the real time requirements of initializing an action in a specific time, and other interrupt problems too. The requirement of high-speed context-sensitive activation of a procedure instead of rule activation in our case, is accomplished by using KEE's Active Value, which meets our timing requirements. As for the garbage collection issue, the fact that we do not use rules at all makes the problem of recycling of no-longer-needed memory elements (of the rule system) non-existent in our case. On the other hand, we utilize the resource facility to recycle our window objects and process objects and use list surgery, if possible, to keep down the rate of generating garbage. It seems that the *SLED* system does not get deteriorated by the Lisp-machine's garbage collection activity.

We depend on the mouse and menu system for explicit interactive operator control. Actually, the normal operator control functions are organized in the "Tools" window. By mouse clicking on an entry of the "Tools" window, the operator can interactively input stored command sequences to the *SLED* system. This organization is made possible by the fact that the mouse process of the Lisp-machine has a higher priority than other processes. Therefore, the system scheduler gives priority to the mouse process, hence it embeds user defined functions for mouse clicks.

Three Layer System Design

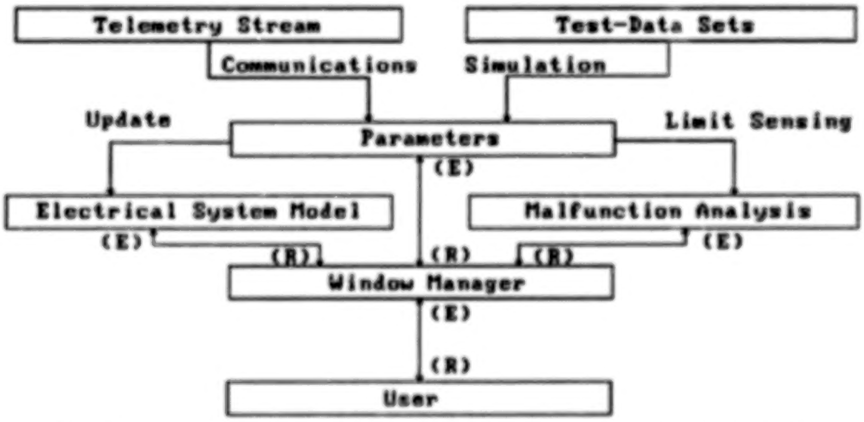
The parameters currently monitored by the *SLED* system are the parameters that appear in the DC distribution drawing and the AC distribution drawing of the Spacelab Systems Handbook. They include: parameters that trigger malfunction analysis procedures, parameters that are referenced by a malfunction analysis procedure, and parameters that indicate an isolated failure.

Object-oriented programming paradigm is used to simplify the design and the implementation of *SLED*. The major functional blocks of the diagnostic task have been turned into objects. These objects model the real world ele-


Telemetry Stream
Communications
Data Distribution
Limit Sensing
Malfunction Analysis
Model Update
Window Manager
User

Down arrows show how data flows during normal operation.
Up arrows indicate re-configuration capabilities.

Figure 3. Functional Blocks Diagram.


Telemetry Stream
Test-Data Sets
Communications
Simulation
Update
Limit Sensing
Parameters
(E)
Electrical System Model
Malfunction Analysis
(E)
(R)
(R)
(R)
(E)
Window Manager
(E)
(R)
User

R: Down arrow shows the Reporting in normal operation
E: Up arrow shows the Editing path in re-configuration

Figure 4. Object Blocks Diagram.

BEST COPY AVAILABLE

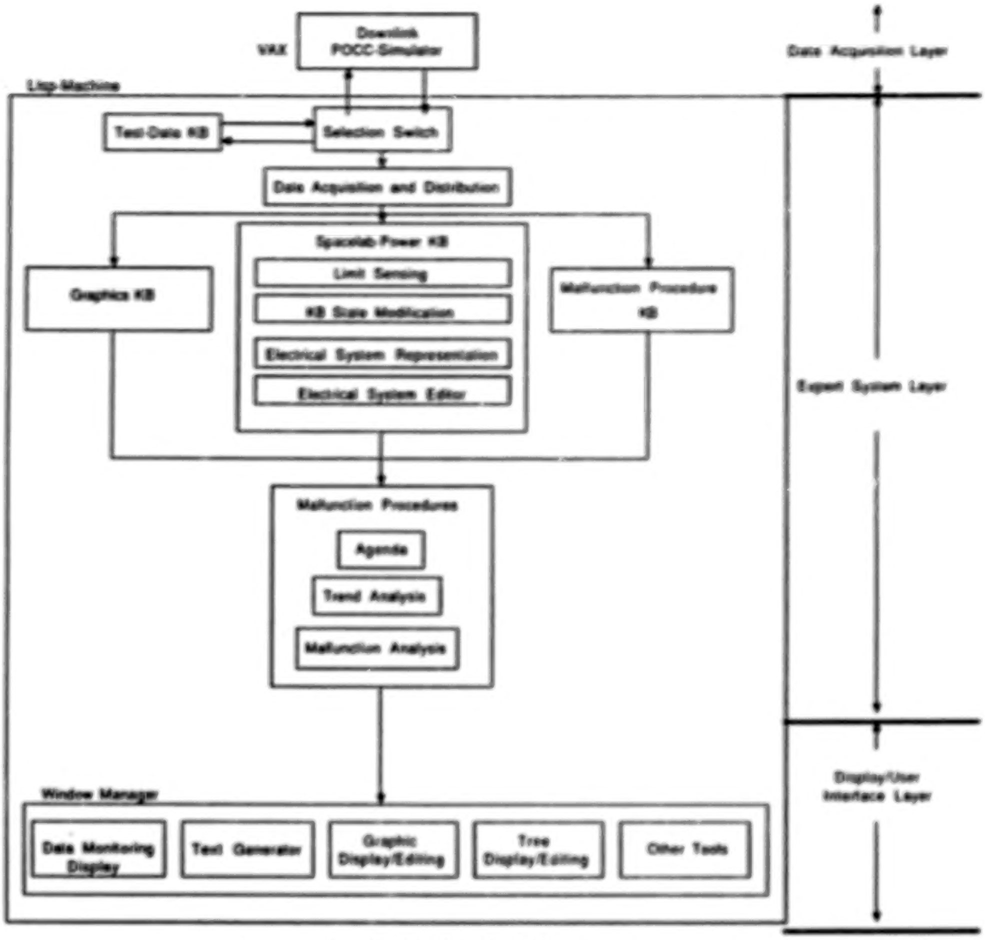

VAX
Downlink
POCC-Simulator
Lisp-Machine
Test-Data KB
Selection Switch
Data Acquisition and Distribution
Spacelab Power KB
Limit Sensing
KB State Modification
Electrical System Representation
Electrical System Editor
Graphics KB
Malfunction Procedure KB
Malfunction Procedures
Agenda
Trend Analysis
Malfunction Analysis
Window Manager
Data Monitoring Display
Text Generator
Graphic Display/Editing
Tree Display/Editing
Other Tools
Data Acquisition Layer
Expert System Layer
Display/User Interface Layer

Figure 5. Three Layer System Design

ments which are relevant to the problems to be solved. Interfaces between functional blocks are thus limited to a few interactions in the form of message passing. The control structure is thus decentralized as an outcome of the object-oriented programming practice. Knowledge has been partitioned in different knowledge bases, which also adds a degree of specialization to all the functions included in every KB. A functional block diagram (Figure 3) and an object block diagram (Figure 4) of the target system help explain the structure of the *SLED* software.

As pointed out in [Strandberg, Abramovich, & etc., 1985], the layered-structure view offers a good organization for discussing system design. We can view the *SLED* system as a three layer structure (Figure 5). The first layer is the Data Acquisition Layer, the second is the Expert System Layer, and the third is the Display/User Interface Layer.

Data Acquisition Layer

As described in [LS-50044-A] and [SLP/2104], the Spacelab scientific data is routed through the subsystem and experiment I/O units, the 192 kb/s telemetry channel, composed of Orbiter and Spacelab data, is available and split up into: two voice channels at 32 kb/s, Orbiter-telemetry data at 64 kb/s nominal, and Spacelab data from experiment and subsystem I/O unit outputs at 64 kb/s nominal. This telemetry channel is software controlled through the Pulse Code Modulation Master Unit (PCMMU). It acquires the data from different sources (Orbiter General Processing Computer, subsystem I/O, and experiment I/O) in a demand and response manner. The Orbiter telemetry data will not need 64 kb/s all the time, so it might be possible that the subsystem and experiment data can be transmitted at a higher rate than 64 kb/s via this telemetry channel. The PCMMU can request data from the Command and Data Management System (CDMS) computers up to 2000 times per second. Upon one of these requests up to 10 data words can be transferred.

Controlled by the Network Signal Processor from the PCMMU, the 192 kb/s telemetry channel is transmitted to ground either via Space Tracking and Data Network (STDN) to the appropriate STDN ground station or via Tracking and Data Relay Satellite System (TDRSS) Ku-Band to the TDRSS ground station.

The Ku-Band data stream is down linked to White Sands, where the data are relayed to the JSC Mission Control Center (MCC). The digital data are recorded and at the

BEST COPY AVAILABLE

same time being fed to a High Rate Demultiplexer (HRDM) in the Payload Operations Control Center (POCC). From the HRDM, real-time data streams are directly fed into the POCC Data Select Switch (PDSS) for distribution.

The data acquisition layer will be resident in the POCC's VAX computer to distribute the synchronized telemetry data to the Expert System Layer resident in the Lisp-Machine via DECnet connection every 3 seconds. Currently, we have not implemented the Data Acquisition Layer. Instead, we are using a set of hand- crafted test-data cases located in the Test-Data KB, along with two synchronized processes: data acquisition process and run sequence process in the LISP-Machine to simulate the Data-Acquisition Layer.

Expert System Layer

The Expert System Layer consists of the KBs and the inference mechanism of using the KBs. We are utilizing IntelliCorp's expert system shell, Knowledge Engineering Environment (KEE), to implement this layer.

There are four KB's in the *SLED* system: the Spacelab-Power KB, the Graphics KB, the Procedures KB, and the Test-Data KB. The Graphics KB is simply filled with run-time information for the application to perform some graphical editing and graphical display operations to support the Display/User Interface Layer. The Procedures KB is used to store the diagnostic text that is displayed in the Malfunction Window and the SSR text. The Test- Data KB is used to store the hand- crafted test data to simulate the Data-Acquisition Layer. This KB is temporary in nature and will be eliminated when the POCC Simulator or the Data-Acquisition Layer is available for generating testing data.

The Spacelab-Power KB is the main KB in the *SLED* system. It changes constantly. It contains the input parameters, the electrical system representation, and some data for the control structure. The representation is based

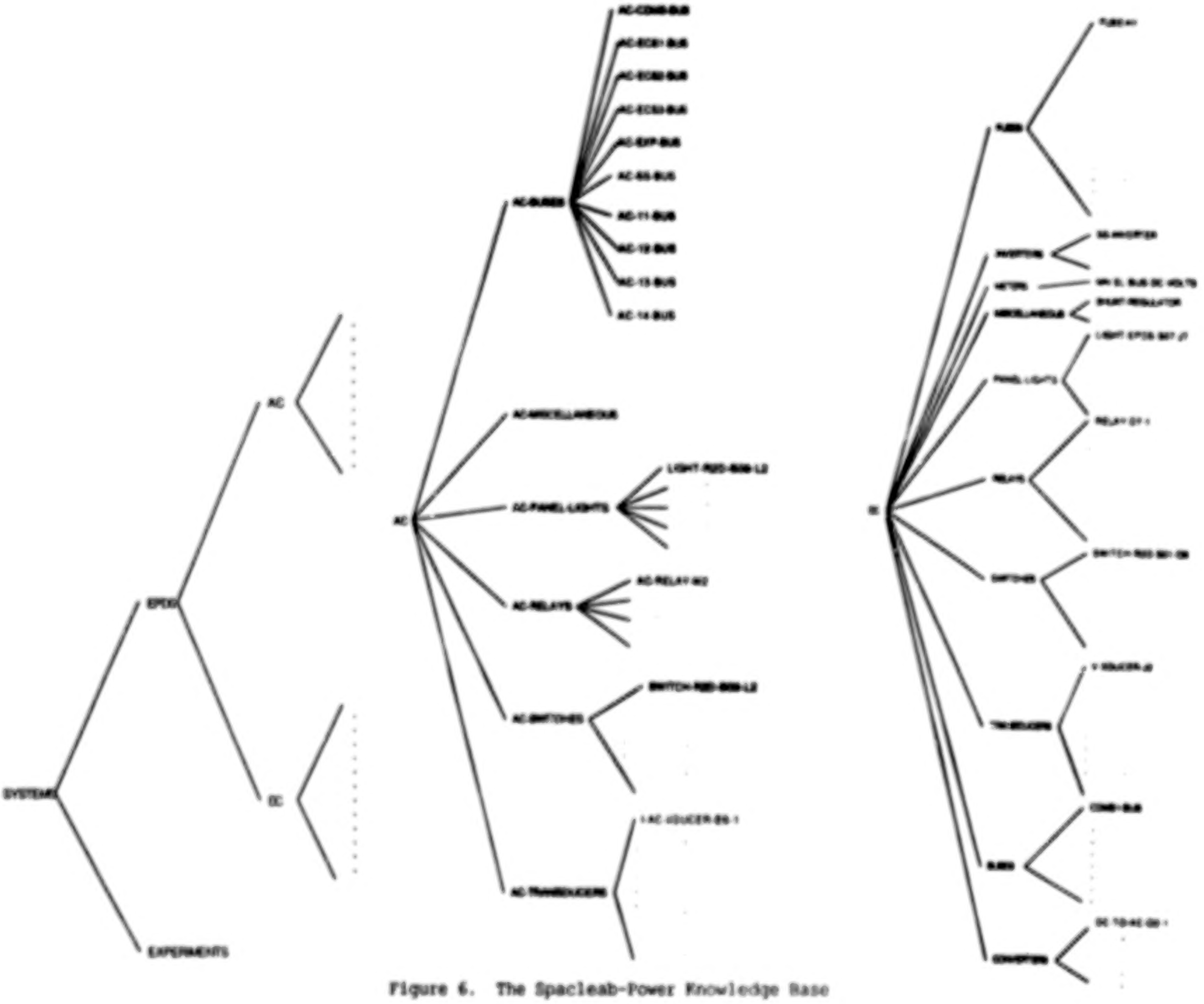

SYSTEMS
EPDS
AC
DC
EXPERIMENTS
AC-BUSES
AC-11-BUS
AC-12-BUS
AC-13-BUS
AC-14-BUS
AC-MISCELLANEOUS
AC-PANEL-LIGHTS
AC-RELAYS
AC-SWITCHES
AC-TRANSDUCERS
FUSES
INVERTORS
METERS
MISCELLANEOUS
PANEL-LIGHTS
RELAYS
SWITCHES
TRANSDUCERS
BUSES
COMM-BUS
CONVERTERS

Figure 6. The Spacleab-Power Knowledge Base

BEST COPY AVAILABLE

on the DC Distribution drawing and the AC distribution drawing of the Spacelab Systems Handbook. It is a perfect copy of these diagrams except for: the diodes are not represented and the return lines are not modeled. The Spacelab-Power KB contains a taxonomy of the objects to model the Spacelab Electrical System, including: Relays, Fuses, Switches, Lights, Meters, Buses, DC-to-AC Converters, AC-to-DC Converters, Inverters, and Transducers. Each electrical element is defined by the actions it can perform, its possible status, its current status, its possible connection, its current connections, and its drawing location. The actions it can perform are methods to which other elements or parameters send messages. This generates an object oriented environment where any element can send an electrical command to any other element without knowing anything about it. Usually the current connections define a message path to propagate the effects. Input parameters are attached to elements. These elements (objects) utilize KEE's Active Value mechanism as a watch-dog for detecting the out-of-limit value. When an out-of-limit value is detected, the appropriate malfunction procedure is activated. Figure 6 shows the taxonomy of the electrical elements of the Spacelab-Power KB.

The Procedures KB is used to store the diagnostic text that is displayed in the Malfunction Window and the SSR procedure that is displayed in the SSR Window. A procedure unit is composed of an active slot, a text slot, and several section slots. The active slot is used by the malfunction procedure kick-off mechanism to determine whether the malfunction has already been fired or not. The text slot, splices the choirs of text that the malfunction analysis needs after every partial conclusion. The section slots contain the malfunction procedure's partial conclusion text. Every section slot corresponds to a node in the malfunction procedure analysis tree (Figure 7) defined by the malfunction procedure structures. The malfunction procedure structure is itself a frame structure.

Display/User Interface Layer

The Display/User Interface Layer uses the knowledge available in the Graphics KB, Procedures KB, and the Spacelab- Power KB to handle all the displaying tasks in the user- interface. We will describe the Windowing and Menu system first.

(i) SLED Window-Frame

The windows and menus used in the *SLED* system are implemented through TI's extension of Common LISP with Window Flavors. Figure 8 shows the main system window. It is divided into six major sub-windows. The two windows in the upper left hand corner are the system time clock, labeled JULIAN, and the mission elapsed time clock, labeled MET. These times are used by the system for reference against timelines and schedules that affect the diagnostics that it performs. In addition, they are used to time-stamp the automatic log entries that the system makes. The time clock also controls the synchronized data acquisition cycle, in the Data Acquisition Layer.

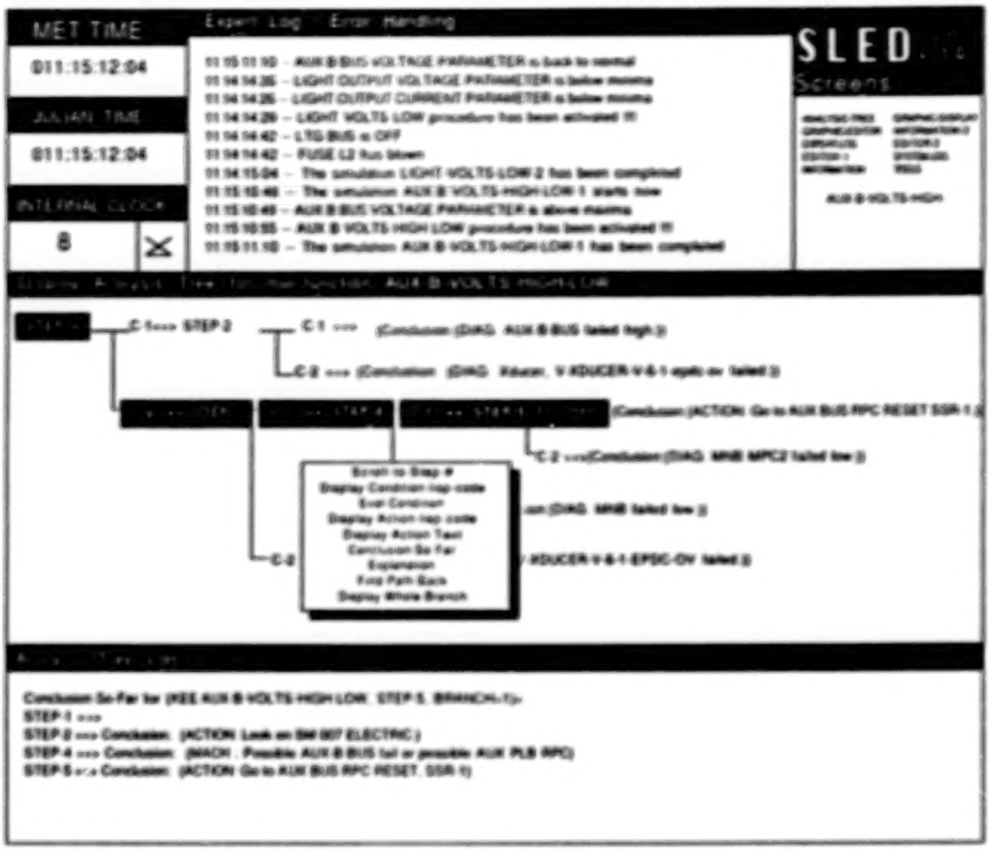
The log of information pertaining to the onboard electrical systems and user actions is located in the top center portion of Figure 8 and is labeled "Expert Log". This log displays the latest information about the status of the electrical systems on board. Each entry is time-stamped and a file containing a complete set of all entries is built for reference to previous events. The log is capable of displaying the last ten messages at all times.

The window at the right top corner of Figure 8, labeled "Screens", is a menu of the various screen configurations available for display. When malfunctions occur, a new entry will appear in this window and can be moused on to be displayed. Among these entries are "Graphic-Editor", "Tools", "Information", "Analysis-Tree", and "Graphic-Display". We will explain some of these screen later.

The thin window approximately one third of the way to the bottom of Figure 8, labeled "User Input", is used for the system to inquire information from the user. At times, all of the information required to make an accurate determination of a problem's cause is not contained in the telemetry downlink. In most instances such as this, a "yes" or "no" query will appear in this window. A positive response is made by clicking the mouse on the query. Not clicking on the query, within a specific period of time, communicates a negative response. At other times, the user may be asked to perform certain functions. A mouse click on the request is taken as confirmation that the action has been taken.

The bottom portion of the screen is the place for displaying different window configurations, selectable from the "Screens" window mentioned above. When the "Tools" entry in the "Screens" window is mouse-selected, or when the system is newly started up, the "Tools" window is displayed, (Figure 8). The "Tools" window is for accessing the set of utilities supplied for the manipulation of the system. Tools are supplied to modify the diagnostic algorithms, display various information about the status of electrical system hardware, display graphic representations of the electrical systems model, build test data sets. display logged information, change the systems status of various electrical system hardware, display SSR procedures, input the text for procedures, display graphic representation of the malfunction diagnostic trees, and display a trace of what caused a certain malfunction conclusion to be reached.

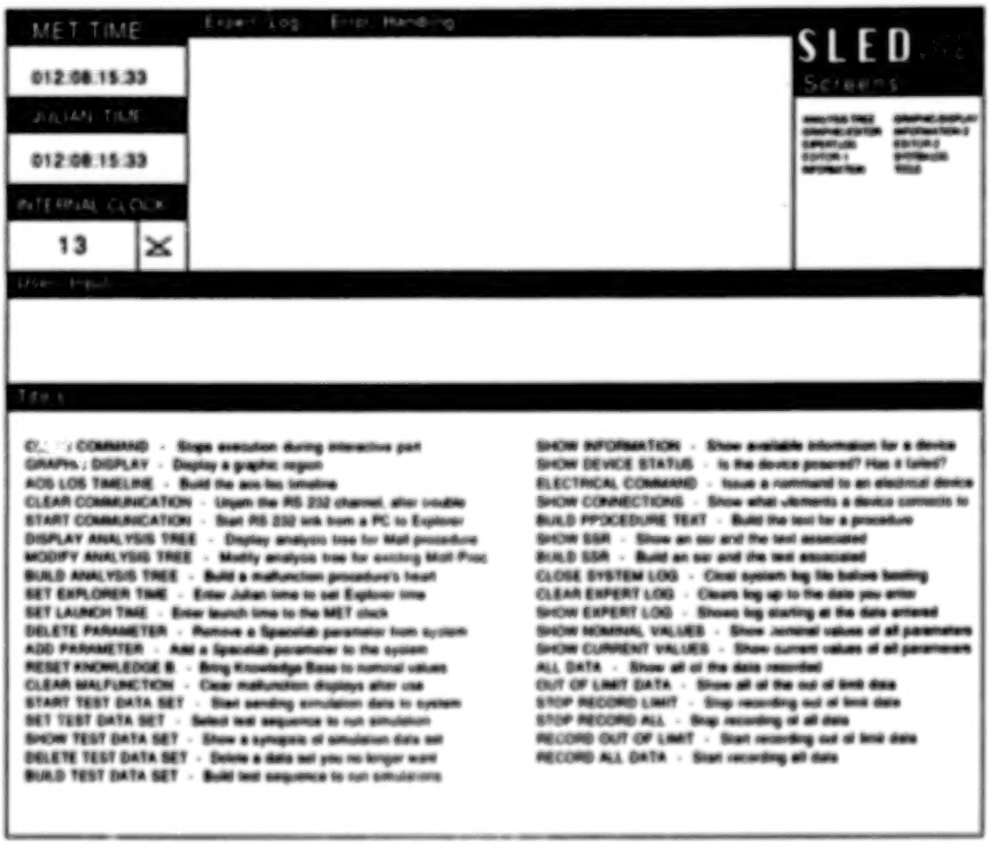
Each configuration has a set of functions associated with it. For instance, the Graphic-Editor configuration is used to build or edit the graphic representation of the electrical system model. When it is the current configuration, the graphic representation of the electrical systems can be modified, as can the electrical system model that the expert


MET TIME
011:15:12:04
JULIAN TIME
011:15:12:04
INTERNAL CLOCK
8
Expert Log - Error Handling
SLED
Screens
AUX B VOLTS HIGH
Diagnostic Analysis Tree for function: AUX B VOLTS HIGH/LOW
STEP 1
C 1==> STEP 2
C 1 ==> (Conclusion (DIAG AUX B BUS failed high))
C 2 ==> (Conclusion (DIAG Xducer, V XDUCER-V-6-1 epdc ov failed))
STEP 3
STEP 4
STEP 5
(Conclusion (ACTION Go to AUX BUS RPC RESET SSR-1))
C 2 ==>(Conclusion (DIAG MNB MPC2 failed low))
on (DIAG MNB failed low))
C 2
V XDUCER-V-6-1-EPDC-OV failed))
Scroll to Step #
Display Condition lisp code
Eval Condition
Display Action lisp code
Display Action Text
Conclusion So Far
Explanation
Find Path Back
Display Whole Branch

```
11:15:11:10 -- AUX B BUS VOLTAGE PARAMETER is back to normal
11:14:14:26 -- LIGHT OUTPUT VOLTAGE PARAMETER is below minima
11:14:14:26 -- LIGHT OUTPUT CURRENT PARAMETER is below minima
11:14:14:29 -- LIGHT VOLTS LOW procedure has been activated !!!
11:14:14:42 -- LTG BUS is OFF
11:14:14:42 -- FUSE L2 has blown
11:14:15:04 -- The simulation LIGHT VOLTS LOW 2 has been completed
11:15:10:46 -- The simulation AUX B VOLTS HIGH LOW 1 starts now
11:15:10:49 -- AUX B BUS VOLTAGE PARAMETER is above maxima
11:15:10:55 -- AUX B VOLTS HIGH LOW procedure has been activated !!!
11:15:11:10 -- The simulation AUX B VOLTS HIGH LOW 1 has been completed
```

Conclusion So Far for (KEE AUX B VOLTS HIGH LOW STEP 5 BRANCH-1)
STEP 1 ==>
STEP 2 ==> Conclusion: (ACTION Look on SM 007 ELECTRIC)
STEP 4 ==> Conclusion: (MACH : Possible AUX B BUS fail or possible AUX PLB RPC)
STEP 5 ==> Conclusion: (ACTION Go to AUX BUS RPC RESET SSR 1)

Figure 7. Diagnostic Analysis Tree Display.

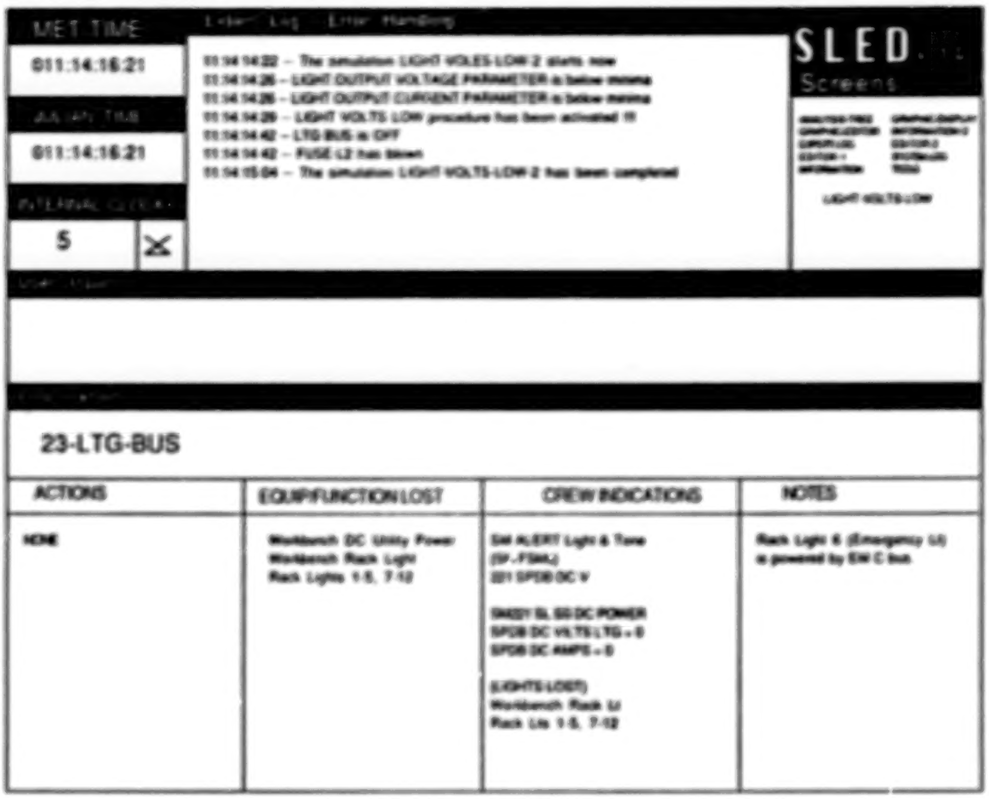

MET TIME
012:08:15:33
JULIAN TIME
012:08:15:33
INTERNAL CLOCK
13
Expert Log - Error Handling
SLED
Screens
User Input
Tools

CLIPS COMMAND - Stops execution during interactive part
GRAPHIC DISPLAY - Display a graphic region
AOS LOS TIMELINE - Build the aos los timeline
CLEAR COMMUNICATION - Unjam the RS 232 channel, after trouble
START COMMUNICATION - Start RS 232 link from a PC to Explorer
DISPLAY ANALYSIS TREE - Display analysis tree for Malf procedure
MODIFY ANALYSIS TREE - Modify analysis tree for existing Malf Proc
BUILD ANALYSIS TREE - Build a malfunction procedure's heart
SET EXPLORER TIME - Enter Julian time to set Explorer time
SET LAUNCH TIME - Enter launch time to the MET clock
DELETE PARAMETER - Remove a Spacelab parameter from system
ADD PARAMETER - Add a Spacelab parameter to the system
RESET KNOWLEDGE B. - Bring Knowledge Base to nominal values
CLEAR MALFUNCTION - Clear malfunction displays after use
START TEST DATA SET - Start sending simulation data to system
SET TEST DATA SET - Select test sequence to run simulation
SHOW TEST DATA SET - Show a synopsis of simulation data set
DELETE TEST DATA SET - Delete a data set you no longer want
BUILD TEST DATA SET - Build test sequence to run simulations

SHOW INFORMATION - Show available information for a device
SHOW DEVICE STATUS - Is the device powered? Has it failed?
ELECTRICAL COMMAND - Issue a command to an electrical device
SHOW CONNECTIONS - Show what elements a device connects to
BUILD PROCEDURE TEXT - Build the text for a procedure
SHOW SSR - Show an ssr and the text associated
BUILD SSR - Build an ssr and the text associated
CLOSE SYSTEM LOG - Close system log file before booting
CLEAR EXPERT LOG - Clears log up to the date you enter
SHOW EXPERT LOG - Shows log starting at the date entered
SHOW NOMINAL VALUES - Show nominal values of all parameters
SHOW CURRENT VALUES - Show current values of all parameters
ALL DATA - Show all of the data recorded
OUT OF LIMIT DATA - Show all of the out of limit data
STOP RECORD LIMIT - Stop recording out of limit data
STOP RECORD ALL - Stop recording of all data
RECORD OUT OF LIMIT - Start recording out of limit data
RECORD ALL DATA - Start recording all data

Figure 8. SLED Main Window with the Tools Screen Displayed

BEST COPY AVAILABLE

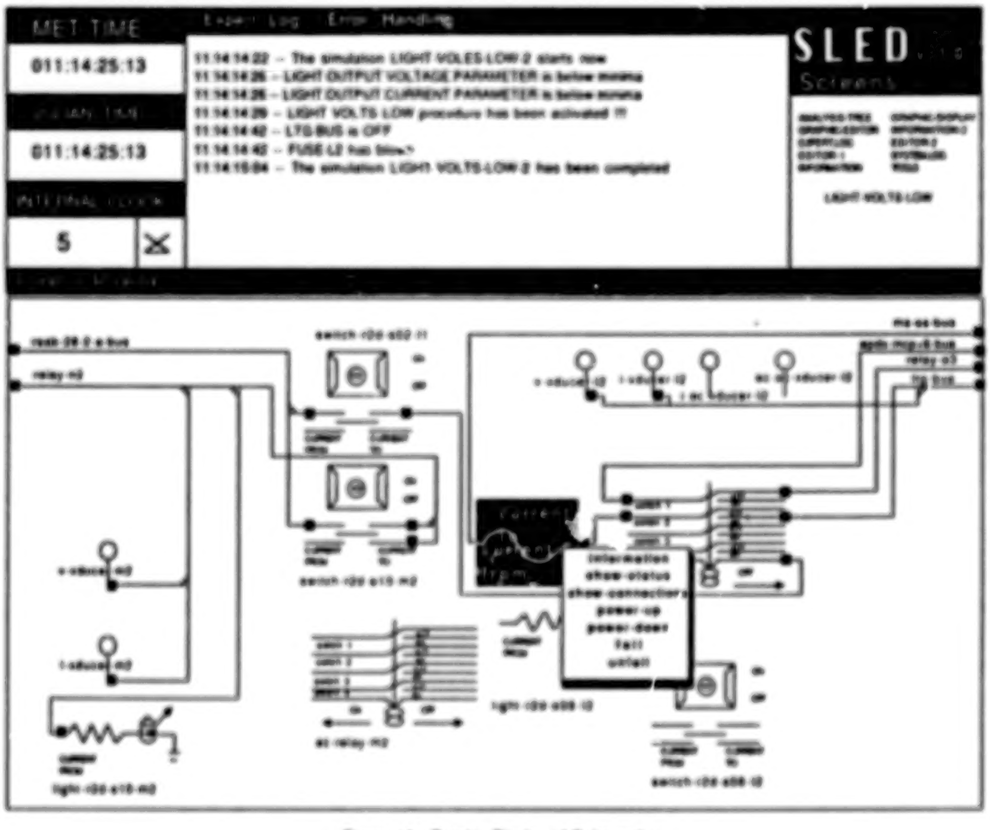

MET TIME
011:14:16:21
JULIAN TIME
011:14:16:21
INTERNAL CLOCK
5
Expert Log – Error Handling
SLED
Screens
LIGHT-VOLTS-LOW

```
11:14:14:22 – The simulation LIGHT VOLES-LOW-2 starts now
11:14:14:26 – LIGHT OUTPUT VOLTAGE PARAMETER is below minima
11:14:14:26 – LIGHT OUTPUT CURRENT PARAMETER is below minima
11:14:14:29 – LIGHT VOLTS LOW procedure has been activated !!
11:14:14:42 – LTG BUS is OFF
11:14:14:42 – FUSE L2 has blown
11:14:15:04 – The simulation LIGHT VOLTS-LOW-2 has been completed
```

23-LTG-BUS

ACTIONS	EQUIP/FUNCTION LOST	CREW INDICATIONS	NOTES
NONE	Workbench DC Utility Power Workbench Rack Light Rack Lights 1-5, 7-12	SM ALERT Light & Tone (SF-FSML) 221 SPD8 DC V SM221 SL SS DC POWER SPD28 DC VLTS LTG = 0 SPD8 DC AMPS = 0 (LIGHTS LOST) Workbench Rack Lt Rack Lts 1-5, 7-12	Rack Light 6 (Emergency Lt) is powered by EW C bus.

Figure 9. SSR Procedure Display


MET TIME
011:14:25:13
JULIAN TIME
011:14:25:13
INTERNAL CLOCK
5
Expert Log – Error Handling
SLED
Screens
LIGHT-VOLTS-LOW
information
show-status
show-connectors
power-up
power-down
fail
unfail

```
11:14:14:22 – The simulation LIGHT VOLES-LOW-2 starts now
11:14:14:26 – LIGHT OUTPUT VOLTAGE PARAMETER is below minima
11:14:14:26 – LIGHT OUTPUT CURRENT PARAMETER is below minima
11:14:14:29 – LIGHT VOLTS LOW procedure has been activated !!
11:14:14:42 – LTG BUS is OFF
11:14:14:42 – FUSE L2 has blown
11:14:15:04 – The simulation LIGHT VOLTS-LOW-2 has been completed
```

Figure 10. Graphic Display of Schematics

BEST COPY AVAILABLE

system uses to perform its diagnosis. The functions associated with the different mouse buttons are displayed at the bottom of the Explorer screen. The mouse buttons change from configuration to configuration but certain conventions have been adhered to wherever possible to make the operation of the system as consistent as possible.

The "Information" screen, "Graphic-Display" screen and the "Analysis-Tree" screen are used as buffers (they are also used for other purposes) to display malfunction-related information. When a malfunction window is visible (Figure 2) and a mousable item is clicked on, then these screens listed above are utilized. If the mousable item is an SSR item, then the "Information" screen is used to display the SSR-procedure, (Figure 9). If the mousable item is an electrical component then the "Graphic-Display" screen is used to display the graphic region, (Figure 10). If the mousable item is the "WHY ?", then the "Analysis-Tree" screen is used to display the diagnosis tree with the diagnosis path highlighted, (Figure 7). Each of these screens are also mouse- sensitive and supports further information querying about each electrical component and each diagnosis step. The mouse click is monitored by the main SLED-Window-Frame process for the window configuration control and the handling of mouse-sensitive items. Three subprocess: Display-Information-Process, Display-Graphics-Process, and Display-Tree-Process are used to display the different type of information (text, schematics, and diagnosis tree) into the appropriate windows.

(ii) Graphical Editor

In order to support the initial input and reconfiguration update of the schematics, a graphical editor is supported.

The graphical editor is one of the most complex features of the system. It is designed to make input of schematic drawings of the Spacelab electrical system a manageable task. It not only makes input of the drawings easier but also makes cross checks to insure that the graphics are accurate representations of the model used by *SLED* in its diagnostics. The graphics for SLS-1 have already been defined and input to the system.

The graphical editor is invoked by moving the mouse into the screens window and clicking on "Graphic-Editor". There are command regions defined by the boxes at the top of the graphic-editor window. Explanations of the uses of some of the commands follow:

choose-an-item: This is used to draw an image of the item within the region specified earlier. The devices that belong in each region have been predetermined by the region specified with which they have been named. Only devices that have been defined in the SPACELAB-POWER knowledge base will be available for graphic representation.

choose-a-drawn-item: This function is used to change the positions of items that have already been drawn. When this function is invoked by a left mouse click, a menu will appear giving as choices the item that have been previously drawn using the choose-an-item function. A left click on one of the items will cause that item to be erased from the position where it was drawn and redrawn where the mouse is when a middle click follows.

connections: The software provided for making the connections between items that are drawn in a graphic region are quite extensive. When invoked, the connections function will display a menu of the items drawn as choices for making connections to or from the item. Choosing an item will cause a menu to pop-up consisting of possible connection points for that item. This list of choices is build from the connections that are defined in the Spacelab-Power knowledge base. In addition to the possible connection points, choices also exist for adding, modifying, or deleting connections in the Spacelab-Power knowledge base. When a connection is made in the system, a box will appear at the connection points of the two items involved. To complete the connection image with wiring, left click where a wire is to terminate at one end. The next middle click will cause a wire to be drawn to the position of the mouse when the middle click is made.

(iii) Build-Analysis-Tree Tool

There are tools for building, modifying, and displaying the malfunction procedure analysis tree to support the reconfiguration capability. The "build-analysis-tree" tool allows the user to create his own analysis procedure. This tool is prompt and menu driven to make it easy for the user to build the malfunction procedure. The system first prompts the user for the malfunction name, comments, time delay, transient spike condition, notification message text, and number of steps involved in the diagnosis procedure. Then a session of menu prompts for conditions and actions to be performed in each diagnostic step are presented. Parameters to be monitored within this malfunction procedure are then solicited by the system with all available parameters being displayed and will then let the user make choices and specify the upper and lower limit condition. Logical and numerical operators are available from the menus to create complicated conditions for testing in each diagnostic step. The action part of the diagnostic step is then requested by the system with menu prompts allowing the user to make choices from those menus. Among those possible actions include: changing the status of any electrical element by sending a message, query the user for input, schedule a task to run in some specific time from the current time, build the text for the current diagnostic step to be displayed in the malfunction window, set the next diagnostic step, and stop analysis, which tells the system that a conclusion has been reached. After the user completes the build analysis tree session, the malfunction procedure just built will be displayed as a diagnostic tree and the user will have the option of modifying it or saving and defining it to the system. If the user selects the define/save option, the newly built malfunction procedure will be activated if the input parameter violates the value range specified by the user in the build-analysis-tree session. That is, the malfunction procedure and the monitoring mechanism is automatically set up in the build-analysis-tree tool.

(iv) Other Tools

There are other useful tools available in SLED. It provides a log for the status of the out-of-limit conditions along with a date and time stamp. The user may look at this log at any time. The log may also be printed, since all of the logged entries are saved to a file. The user can view the current and nominal values of the parameters that the system looks at. He can obtain the status of a particular electrical component. He may also see what other components are connected to it along with specific information about the component. One of the major capabilities of this system is that it allows the user, a non-programmer, to create a malfunction recovery procedure, should the need arise. The different tools that allow the user to do this include building the diagnostic tree (described above), building the procedure text, the text that is displayed in the unique malfunction recovery window, building the SSR text, the text that is viewed for the recovery procedure, and adding the necessary parameters needed to diagnose the malfunction. The user also has the capability of deleting, modifying, or viewing the different parts of the malfunction recovery procedure. This includes modifying or viewing the malfunction recovery procedure, viewing the SSR text, and deleting or viewing the parameters. The user may also set up an AOS-LOS timeline. This is used to "wake-up" the system during AOS, and put the system to "sleep" during LOS. In order to test the *SLED* system, tools were incorporated into the system to test its reliability. These tools includes building, viewing, deleting, and setting the test data sets.

In addition to these tools, specifically designed for *SLED*, the standard user-interface tools of KEE are always available for the advanced user in data monitoring or KB updates. For example, we can use the KEE user-interface for parameter monitoring and trend analysis.

4. CURRENT STATUS

Testing

The validation process at the current time is not formal. Three levels of testing are planned to be performed: 1. Hand- crafted simulation data, within the Explorer; 2. POCC simulator or tape of real data with Explorer to VAX communications. 3. Real-Time operational test during the SLS-1 mission. We have used hand-crafted test-data stored as KEE's units and user interaction to aid testing. We plan to use the POCC simulator and taped records of real data to test the system. Due to the complexity and dynamic nature of the problem domain, the real system must be verified by actual on-line testing during the SLS-1 mission. We plan to have the *SLED* system in operation, in parallel to the conventional system during the SLS- 1 mission scheduled for June 1990.

During the SLS-1 mission, it is planned to support the *SLED* system in parallel with the current screen displays and monitoring tools of the POCC operation facility. The *SLED* terminal will be located next the the existing POCC electrical power monitoring terminal. This arrangement will also show the capability of *SLED* to the PSE and enhance the user acceptance of the *SLED* expert system.

While the testing of *SLED* with the simulator and the final testing of the *SLED* system during the SLS-1 mission is still on schedule, evaluation by the domain expert so far has been positive. We will not address the problem of how to measure the performance of *SLED* as a trouble shooting expert system in a formal sense. Instead, the actual comparison of the *SLED* performance with the current monitoring system and the PSE's performance in diagnosis of the fault, is of great interest to us. The *SLED* system (fault) log file will be used as an audit trail to verify the overall system performance against the written fault logs produced by the PSE.

Currently we have finished level 1 testing procedures. The POCC simulation software is currently in development and will be available in August 1989. The hand-crafted test case includes data that fired multiple malfunctions. All the tests that were performed went well and the response time was about 15 seconds in handling a single malfunction. This time was measured from the detection of the out-of-limit value to the completion of the diagnosis of the malfunction, assuming immediate user action for all user input requested. It also takes 15 seconds for the 2 malfunction case, and it takes about 30 to 50 seconds for a 5 malfunction case. The point here is that the time needed for multiple malfunctions is growing linear with the number of malfunctions. In particular it is not increasing exponentially with the number of malfunctions occurring. The reason for this linearity of response time is the way we treat each malfunction independently in a separate window with an independent diagnostic process. No interrelations between the malfunctions are considered in our model. Therefore, the diagnosis process of each malfunction is running in parallel. In theory, the response time of the multiple malfunction case should be about the same as the single malfunction case. But the fact that independent processes are time-sharing processes in the host LISP machine and that there is only one user interface window, makes the total time grow approximately linear with the number of malfunctions occurring.

Enhancement Plan

As mentioned in the Background section, the current *SLED* system is the first phase of the Spacelab electrical power system which contains only the knowledge about the EPDS. No knowledge of the Experiment equipment have been included. We are currently adding the malfunction procedures, the SSR's, and the schematics of the subsystem up to the experiment hardware to the *SLED* system, as required in phase 2, and then, finally adding the knowledge of each experiment equipment to the system (phase 3). Obviously, a lot more knowledge needs to be added to the system. This is making us approach the limitation of the available hardware and software. The total number of frames, in KEE it is referred to as units, is 363 units within the Spacelab- Power KB, 519 units in the Test-Data KB, 125 units in the Graphics KB, and 83 units in the Procedures KB. The total number of experiment equipments that we would like to represent is 40. For each experiment we need to increase the Graphics KB by approximately 10 units and the Spacelab-Power KB by approximately 50 units. Besides, there is a lot of disk space that is used to store the bit-maps for the schematics, 4.7 MB of disk space is used for the schematics of the Spacelab EPDS- distribution. It becomes impractical to use the current scheme, i.e. direct bit-map storage to store all the schematics for the experiment equipment to be monitored. A better way of storing the schematics needs to be implemented. The new approach we have in mind is to build up the schematics on-the-fly when requested. The schematics will be built using the electrical component and connection information in the KB's. Of course, there is a trade-off between time and disk and memory space. The on-the-fly approach will slow down the response time for the schematics display. Fortunately, the schematic display is not as time-critical as the monitoring, diagnostic, and malfunction procedure portion. The tremendous disk space saving justifies the slow response time in the schematic display.

The current system is hosted in the Texas Instruments (TI's) Explorer Lisp-Machine. Our target delivery system will be TI's Micro-Explorer, i.e. a Mac II with TI's Lisp processor board. We will start the conversion when our Micro-Explorer system is available, which will be in March of 1989. We expect the conversion effect to be a minimal.

We would like to make the knowledge acquisition tools: the graphical editor, used for creating and updating the schematics, and the diagnostic tree building tools, used to input and update the malfunction procedures, to be more user friendly in the near future, which will incorporate the users suggestions.

We would also like to use rule representations of the diagnosis logic instead of the current explicit representation of the logic. This would be a major overhaul and the task of meeting the real time speed requirements may pose some tough problems, because KEE's rule system is slow.

5. CONCLUSION

We have successfully demonstrated the applicability of expert system technology to the Spacelab power subsystem diagnostic problem. The response time of about 15 seconds, i.e. the time needed for 5 data cycles, for a malfunction diagnosis, with the corrective malfunction procedure, SSR procedure and schematics readily available, well exceeds the performance of a trained PSE. Besides the on-line storage and fast retrieval of the schematics and the availability of status information for each electrical component will make the system a big help for the PSE in dealing with the malfunctions which are not covered in the malfunctions procedure handbook.

With our experience of the development of the *SLED* system, there are several important points in real time expert system development we would like to re-iterate:

1. To narrow down the application domain is a very important design issue. After actually trying to solve a more general problem, the *SLED* system evolved from the General Diagnostic System. This point has been emphasized so many times in the literature, e.g. [Hayes-Roth, Waterman & Lenat, 1983], it is too easy to overlook.

2. Expert systems for real time operations is achievable. Besides the planned enhancements, the system can be extended to other areas of spacecraft mission control problems.

3. The first use of *SLED* was for training. As has been pointed out in the literature, e.g. [Buchanan & Shortliffe, 1984], the explanation capability makes expert systems an excellent tool for training.

4. The system can be used as a validation and verification tool for the user to generate the malfunction procedures. It can be extended to a tool for generating the malfunction procedure document.

5. When the next generation of Lisp-machines based on a Lisp chip is stabilized, it will be feasible to put the Lisp-based fault diagnostic expert system on-board Spacelab. Actually, this approach would be simpler and may speed up the fault diagnostic task of the experiment, because we can further partition the problem and place the expert system hardware and software directly in the relevant area. For example, the *SLED* system can be placed inside the experiments box and does not need to store the system model and knowledge about Spacelab itself.

Acknowledgement

We would like to thank Marshal Space Flight Center (MSFC), Johnson Space Center (JSC), and the Electrical group of the Systems Engineering section of GE Government Services Houston for their support and guidance during this project.

References

[JSC-18927]
Flight Data File, Spacelab Malfunction Procedures, Basic, Revision B, March 1985, Mission Operations Directorate, JSC, NASA.

[JSC-12777C]
Spacelab Systems Handbook, Rev. C, Aug. 1983, Mission Operations Directorate, JSC, NASA.

[LS-50044-A]
Data Requirements for SLS-1 JSC Life Sciences Flight Experiment, pp. 2.2 - 2.5.

[SLP/2104]
Spacelab Payload Accommodation Handbook, Main Volume, issue no: 2, revision: 0, Aug. 1985, J. W. Thomas Manager, NASA MSFC Spacelab Program Office.

[Buchanan & Shortliffe, 1984]
Rule-Based Expert Systems; B. Buchanan, and E. Shortliffe; Addison-Wesley, Reading, MA., 1984.

[Cantone, Pipitone & etc., 1983]
Model-Based Probabilistic Reasoning for Electronic Trouble- Shooting; R.R. Cantone, F.J. Pipitone, W.B. Lander, M.P. Marrone; Preceedings IJCAI 1983.

[D'Ambrosio, Fehling & etc., 1987]
Real-Time Process Management for Materials Composition in Chemical Manufacturing; B. D'Ambrosio, M.R. Fehling, S. Forrest, P. Ranlef, and B. Michael Wiber; IEEE/Expert, Summer 1987.

[Davis, 1983]
Reasoning From First Principles in Electronic Trouble-Shooting; Randall Davis; International Journal of Man-Machine Studies, Vol. 19, 1983.

[Dickey & Toussaint, 1984]
An Application of Expert Systems to Manned Space Stations; F.J. Dickey and A.L. Toussaint; CAIA 1984.

[Forgy, 1981]
OPS-5 User's Manual, Report CMU-CS-81-135; C.L. Forgy, Dept. of CS, Carnegie-Mellon University.

[Geffner & Pearl, 1987]
Distributed Diagnosis of Systems with Multiple Faults; H. Geffner and J. Pearl; CAIA 1987.

[Griesmer, Houg & etc., 1985]
YES/MVS: A Continuous Real-Time Expert System; J.H. Griesmer, S.J. Houg, M. Karnaugh, J.K. Kastner, M.I. Shor; IJCAI 1985.

[Hamilton, 1986]
SCARES-A Spacecraft Control Anomaly Resolution Expert System; M. Hamilton; Expert System in Government Symposium, 1986.

[Hayes-Roth, Waterman & Lenat, 1983]
Build Expert Systems; F. Hayes-Roth, D.A. Waterman, and D.B. Lenat; Addison-Wesley, Reading, MA, 1983.

[KEE 2.1]
IntelliCorp KEE Software Development System User's Manual; July 18, 1985.

[Leinweber, 1987]
Expert Systems in Space; D. Leinweber; IEEE/Expert, Spring 1987.

[Maletz, 1985]
An Architecture for Consideration of Multiple Faults; M.C. Maletz; CAIA 1985.

[Milne, 1987]
Fault Diagnosis Through Responsibility; R. Milne; IJCAI 1987.

[Muratore, Madision & etc., 1988]
Real Time Expert System Prototype for Shuttle Mission Control; J.F. Muratore, R.M. Madison, T.A. Heindel, T.B. Murphy, R.F. McFarland, A.N. Rasmussen, E.D. Kindred, C.L. Whitaker; Telemetry Conference, 1988.

[Pipitone, 1984]
An Expert System for Electronic Trouble-Shooting Based on Function and Activity; F. Pipitone; CAIA 1984.

[Rook & Odubiyi, 1986]
An Expert System for Satellite Orbit Control (ESSOC); F.W. Rook and J.B. Odubiyi; Expert Systems in Government Symposium, 1986.

[Silverman, 1986]
Facility Advisor: A Distributed Expert System Testbed for Spacecraft Ground Facilities; Barry G. Silverman; Expert Systems in Government Symposium, 1986.

[Stallman & Sussman, 1979]
Problem Solving about Electrical Circuits; R.M. Stallman and G.I. Sussman; In Artificial Intelligence: An MIT Perspective, 1979, Edited by P. Winston and R. Brown.

[Strandberg, Abramovich, & etc., 1985]
PAGE-1: A Trouble-Shooting Aid for Non impact Page Printing Systems; C. Strandberg, I. Abramovich, D. Mitchell, K. Prill; CAIA, 1985.

[Taie & Srihari, 1986]
Device Modeling for Fault Diagnosis; M.R. Taie and S. Srihari; Expert Systems in Government Symposium, 1986.

[Wilkinson, Happell, & etc., 1988]
Achieving Real-Time Performance in FIESTA; W. Wilkinson, N. Happell, S. Miksell, and R. Quillin; 1988 Goddard Conference on Space Application of Artificial Intelligence, Goddard Space Flight Center, NASA, May 1988.

BLANK PAGE

SHARP: A MULTI-MISSION AI SYSTEM FOR SPACECRAFT TELEMETRY MONITORING AND DIAGNOSIS

Denise L. Lawson
Mark L. James

Artificial Intelligence Group
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, California 91109

ABSTRACT

The Spacecraft Health Automated Reasoning Prototype (SHARP) is a system designed to demonstrate automated health and status analysis for multi-mission spacecraft and ground data systems operations. Telecommunications link analysis of the Voyager II spacecraft is the initial focus for the SHARP system demonstration which will occur during Voyager's encounter with the planet Neptune in August, 1989, in parallel with real-time Voyager operations.

The SHARP system combines conventional computer science methodologies with artificial intelligence techniques to produce an effective method for detecting and analyzing potential spacecraft and ground systems problems. The system performs real-time analysis of spacecraft and other related telemetry, and is also capable of examining data in historical context.

This paper gives a brief introduction to the spacecraft and ground systems monitoring process at the Jet Propulsion Laboratory (JPL). It describes the current method of operation for monitoring the Voyager Telecommunications subsystem, and highlights the difficulties associated with the existing technology. The paper details the approach taken in the SHARP system to overcome the current limitations, and describes both the conventional and artificial intelligence solutions developed in SHARP.

INTRODUCTION

The Voyager II spacecraft was launched from Cape Canaveral, Florida on August 20, 1977. The technology to track and monitor such a probe was designed and developed in the early 1970's. This now antiquated technology, coupled with the efforts of bright, resourceful scientists, has carried Voyager through near-fatal catastrophic events to three of our solar system's outer planets (four in August). Despite failed radio receivers, sunlight damage to the photopolarimeter scientific instrument, and a partially paralyzed scan platform (which houses Voyager's imaging system), these scientists have kept Voyager operational, enabling the capture and transmission of vast amounts of invaluable information and images of the Jovian, Saturnian, and Uranian systems.

During critical periods of the mission, up to 30 real-time operators are required to monitor the spacecraft's ten subsystems on a 24-hour, 7-day per week schedule. This does not include the numerous subsystem and scientific instrument specialists that must constantly be available on call to handle emergencies.

As more and more solar system explorations are undertaken, it will become increasingly difficult to staff a large enough effort to support these expensive missions. Currently there is one Mission Control Team and one Spacecraft Team for each flight project. JPL has initiated an effort to coordinate all missions through a central Space Flight Operations Center (SFOC) whose goal is to transition from single-project dedicated flight teams to one multi-mission team which flies all spacecraft. Within SFOC, the Voyager spacecraft will continue to be monitored throughout their extended mission of discovering the solar system heliopause (the boundary between the Sun's magnetic influence and interstellar space); the Magellan spacecraft, to be launched in April, 1989, will be tracked throughout its flight to Venus; the Galileo mission to Jupiter will be monitored; and other new flight projects will be observed throughout their operation by this single multi-mission flight team.

The Spacecraft Health Automated Reasoning Prototype (SHARP) is an effort to apply artificial intelligence (AI) techniques to the task of multi-mission monitoring of spacecraft and diagnosis of anomalies. Ultimately, SHARP will ease the burden that multiple missions would inevitably place upon subsystem, scientific instrument, and Deep Space Network (antenna) experts. SHARP will automate many of the mundane analysis tasks, and reduce the number of operators required to perform real-time monitoring activities. The system will enhance the reliability of monitoring operations, and may prevent those types of errors that cause spacecraft such as the Soviet Phobos to be lost.

The Voyager II spacecraft was targeted for the prototype effort since it is currently the only spacecraft in flight which has yet to complete its primary mission. The prototype effort was further focused to one subsystem so that specific concepts could be developed and then demonstrated in a vigorous operational setting: the spacecraft's encounter of the planet Neptune in August, 1989. The Telecommunications subsystem was chosen for the initial demonstration since anomalies occur on a frequent basis in this area, and the Telecom expert, Mr. Boyd Madsen, demonstrated enthusiastic support. The Telecommunications area also presents the challenge of coordinating monitoring and diagnosis efforts of both the spacecraft and ground data systems.

Like many artificial intelligence applications, in order to supply the AI component of a system with real data, a substantial effort must be invested in the development of other aspects of the system. This may entail utilizing standard conventional computer science methodologies and enhanced graphical capabilities. The SHARP system efficiently incorporates these technologies to complement the use of AI techniques.

CURRENT METHOD

In current mission operations practice each spacecraft is monitored daily, and during planetary encounters monitoring is continuous. Three complexes of antennas located around the world comprise NASA's Deep Space Network (DSN), in the Mojave dessert at Goldstone, California; near Canberra, Australia; and near Madrid, Spain. With the exception of occultations and a short gap between the Canberra and Madrid stations, the spacecraft is always in view from one of these Deep Space Stations (DSS). Such a scheduled period of observance of the spacecraft by a DSS is called a pass.

Required Data

In order to effectively analyze the telecommunications link from the spacecraft through the Deep Space Network and ultimately to the computers at JPL, a wide variety of information must be accessed and processed. This analysis occurs in real-time as well as prior to the scheduled spacecraft pass.

Predicts are numerical predictions of acceptable threshold values for particular spacecraft and DSS parameters. The current method of generating pass predicts is by searching large hardcopy listings of raw predicts to find the correct spacecraft, station, time, and other approximated information. Predicts are then manually corrected to reflect the actual spacecraft state using a hand calculator, and the results are manually recorded on a data sheet.

Another piece of information pertinent to spacecraft monitoring is the Integrated Sequence of Events (ISOE). The ISOE is a hardcopy of scheduled spacecraft activity integrated with DSS information. ISOE data is used extensively throughout the monitoring process in predict data, alarm determination, graphics, and diagnosis. The Voyager ISOE must be visually scanned, and Telecom events manually highlighted by the real-time operator so that the Telecom activity can be monitored. A handwritten correction sheet is issued for each modification to an ISOE.

Telemetry data from the spacecraft, tracking stations, and other relevant systems is collected in the JPL computers and separated into channels that are distributed across JPL for processing and analysis. These channels contain the values of hundreds of spacecraft engineering parameters and station performance parameters. The channels are plotted on black and white computer screens and are visually monitored to ensure that they remain within their pre-specified limits.

Also critical to the communications link analysis are alarm limits, the threshold values for spacecraft and DSS performance. These values are selected according to the status of several parameters. However, the process to change these limits is manual and must be performed in real-time. The procedure is so impeditive, and occurs so often, that typically a wide threshold is selected which incorporates the entire range of parameter conditions, creating the risk of undetected anomalies.

Limitations

Due to cumbersome and time-consuming processes, several limitations exist on the current method of analyzing Voyager telecommunications link data.

The tedious manual process for predict generation may take up to two hours each day, and limits calculations to one predict point per hour. Actual link parameters may be received every 15 seconds, leaving quite a disparity between the desired number of predictions and the incoming data.

The Integrated Sequence of Events prompts several complications as well. During periods of heightened activity, it is possible for a single Telecom event to be embedded among several pages of another subsystem's events in the ISOE. It is easy to miss events, and sometimes the ISOE is so extensive that operators do not even attempt to scan it. Rather, they rely on an unofficial graphical sequence hardcopy product, the Spacecraft Flight Operations Schedule (SFOS), to monitor critical events. The SFOS, which is manually highlighted with a marker to indicate changes, creates problems when users unknowingly do not reference the latest activity modifications.

The current Voyager data display system presents another area of limitation. It allows only five plot display pages for the entire spacecraft team. The Telecommunications subsystem has control of a single page. One display page is capable of showing up to three plotted channels. In order to change the plot parameters to select different channels to display, the operator must punch a card and feed it into the system's card reader. To obtain an additional plot, special permission must be secured from personnel of another subsystem who are willing to temporarily give up one of their own plots.

Broadened alarm limits present obvious complications. If, in fact, a component is in alarm within the broadened range, this condition will go undetected. For spacecraft activities which warrant an alarm limit change, and the operator chooses to forego the unwieldy paperwork process, then he must endure the false alarm for the remainder of that spacecraft activity.

A tabular display of spacecraft parameters is available which indicates alarms by reversing the color of the alarmed channel's field. However, this display is seldom used as the operator is usually viewing plotted data on the one allotted Telecom display page. As a result, a Telecom alarm condition generally is not detected by the Telecom operator until the Voyager Systems Analyst (who monitors and coordinates all subsystems) calls it to his attention.

Diagnosis

When a spacecraft or DSS parameter goes into alarm, the cause must be determined. In many cases, the condition is actually a false alarm due to inaccuracies precipitated by the limitations of the system. In other instances, the alarm exists because of common problems that occur on a frequent basis. For actual spacecraft problems, such as the failed radio receiver, hundreds of people must be notified and put on alert to solve the emergency. Regardless of the cause or severity of an alarm, a standard set of rules is routinely followed to determine the basis of the problem. Unfortunately, this knowledge resides with a select few, and the first rule of the standard procedure is to consult the expert, even when the situation arises from a known false alarm.

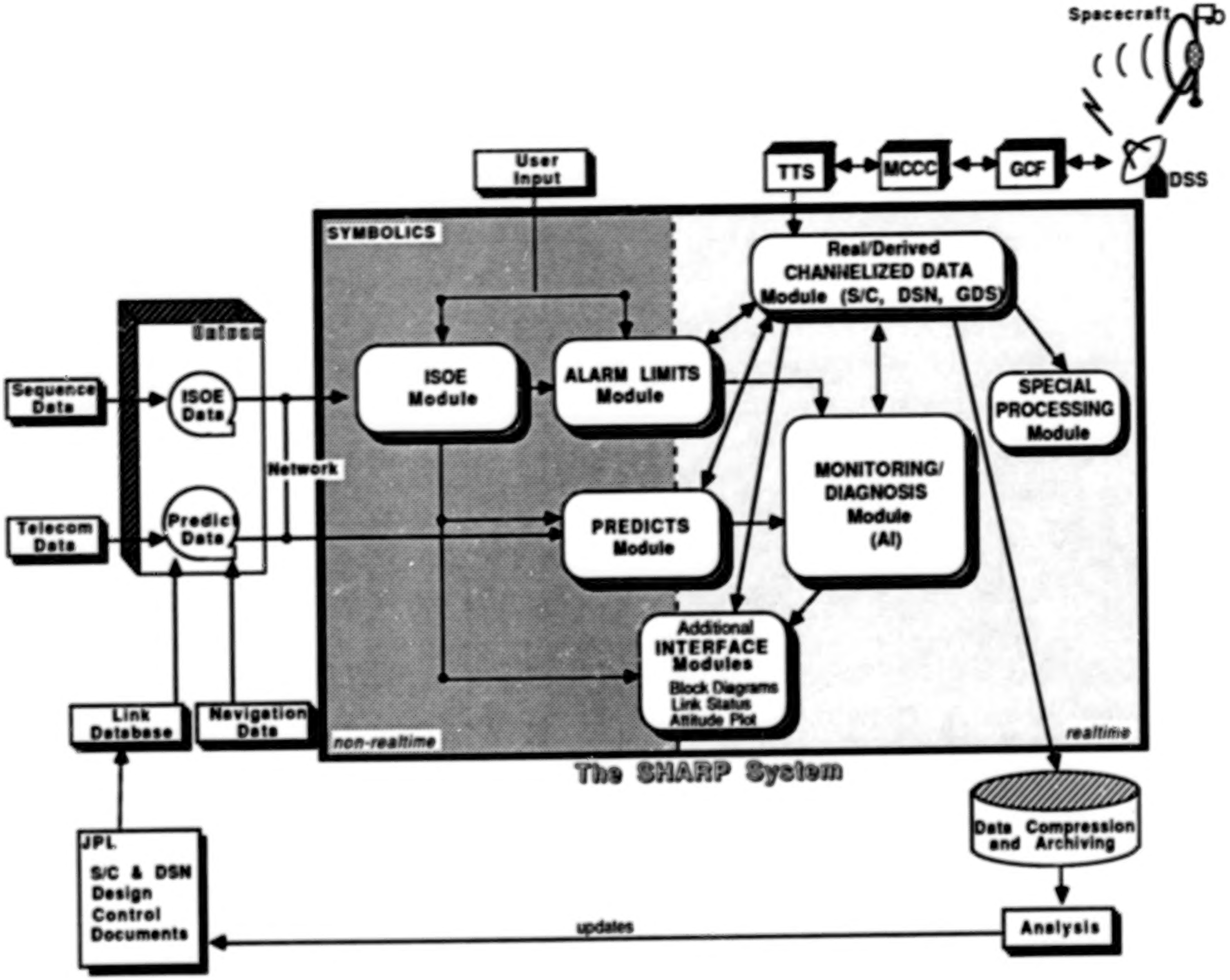
THE SHARP SOLUTION

SHARP introduces automation technologies to the spacecraft monitoring process to eliminate much of the mundane processing and tedious analysis. The SHARP system features on-line data acquisition of all required information for monitoring the spacecraft and diagnosing anomalies. The data is centralized into one workstation and serves as a single access point for the aforementioned data as well as for the diagnostic heuristics. Figure 1 illustrates a top-level view of the SHARP system. Shown are the individual modules that comprise the system, as well as relevant components which are external to SHARP.

SHARP is implemented in CommonLISP on a SYMBOLICS 3650 color LISP machine. Many components of the system utilize STAR*TOOL [1] (patent pending), a language and environment developed at JPL which provides a tool box of state-of-the-art techniques commonly required for building AI systems. The SHARP system currently consists of approximately 40,000 lines of CommonLisp code, and STAR*TOOL comprises an additional estimated 85,000 lines of CommonLisp code.

Conventional Automation

The SHARP system captures raw predicts for on-line storage and processing. When the predicts are generated for the Voyager Spacecraft Team as hardcopy, the information is transferred over an RS-232C serial line to the SHARP system. Pass predicts may then be automatically generated at 15 second intervals, the shortest possible time interval between the arrival of any two spacecraft data points. Instantaneous predicts, which are pass predicts corrected in real-time for spacecraft pointing loss and DSS system



Spacecraft
DSS
User Input
TTS
MCCC
GCF
SYMBOLICS
Database
Sequence Data
ISOE Data
Telecom Data
Predict Data
Network
ISOE Module
ALARM LIMITS Module
Real/Derived CHANNELIZED DATA Module (S/C, DSN, GDS)
SPECIAL PROCESSING Module
MONITORING/ DIAGNOSIS Module (AI)
PREDICTS Module
Additional INTERFACE Modules
Block Diagrams
Link Status
Attitude Plot
Link Database
Navigation Data
non-realtime
realtime
The SHARP System
Data Compression and Archiving
JPL
S/C & DSN Design Control Documents
updates
Analysis

Figure 1. SHARP Telecom System Overview

noise temperature, are also automatically calculated at 15 second intervals. Spacecraft and DSS residuals, difference measurements between the actual values and predicted values, are automatically derived in real-time.

SHARP also acquires the Integrated Sequence Of Events for on-line storage and viewing. A generic capability to extract subsystem specific information has been developed, hence Telecom-specific events may be stripped from the ISOE and displayed to enable rapid identification of significant Telecom activities to be monitored during any particular pass. Editing capabilities facilitate on-line additions, deletions, and other changes to the ISOE, thus reducing the likelihood of referencing outdated material.

New plotting capabilities for the channelized data have also been implemented within the SHARP system. The operator can construct as many data plots per page, or screen, as desired, although five plots per page seems to be the optimal number for effective viewing. The user also possesses the capability to construct multiple pages which can be set as a program parameter. The user can change the display of plots on any given page at any time with simple menu-driven commands.

Alarm tables have also been constructed as part of the conventional automation process, and placed on-line within the SHARP system. A table for each relevant spacecraft configuration exists, resulting in alarm limits representative of the true thresholds for each data channel. SHARP determines alarm limits dynamically in real-time and accurately reflects each spacecraft or DSS configuration change. Dynamic alarm limit determination eliminates the cumbersome alarm change paperwork process as well as many occurrences of false alarms.

Several graphical displays in SHARP automatically highlight alarmed events as they occur. These displays offer information ranging from the location of a problem to the probable cause of the alarm.

Enhanced Graphical Capabilities

The SHARP system provides numerous sophisticated graphical displays for spacecraft and station monitoring. A comprehensive user interface has been developed to facilitate rapid, easy access to all pertinent data and analysis. Displays have been constructed which range from placing data on-line to the creation of detailed graphics which provide a multitude of information at one glance. An interface exists for each major module of the SHARP system. Each interface provides customized functions which allow data specific to that module to be easily accessed, viewed, and manipulated. Each SHARP module can be accessed from any other module at any time, and all displays are in color with mouse sensitivity and menu-driven commands. Figure 2 shows the SHARP top-level system status view.

The Predicts interface in SHARP allows tabular display of raw predicts, pass predicts, instantaneous predicts, and residuals for any specified time range. A color-coded DSS availability graph has also been designed which enables rapid identification of available stations for any given viewing period. Situations which mandate that another Deep Space Station be acquired can hence be addressed immediately as opposed to the more arduous current method which requires the manual look up of each station at the specified time period.

The SHARP system provides an Integrated Sequence of Events interface which offers numerous capabilities to the operator. On-line viewing of any ISOE is available, and intricate modifications may be performed with ease. Editing the ISOE is accomplished via menu-driven commands which contain explanations of the complex ISOE data. For example, CC3A32330 means that the

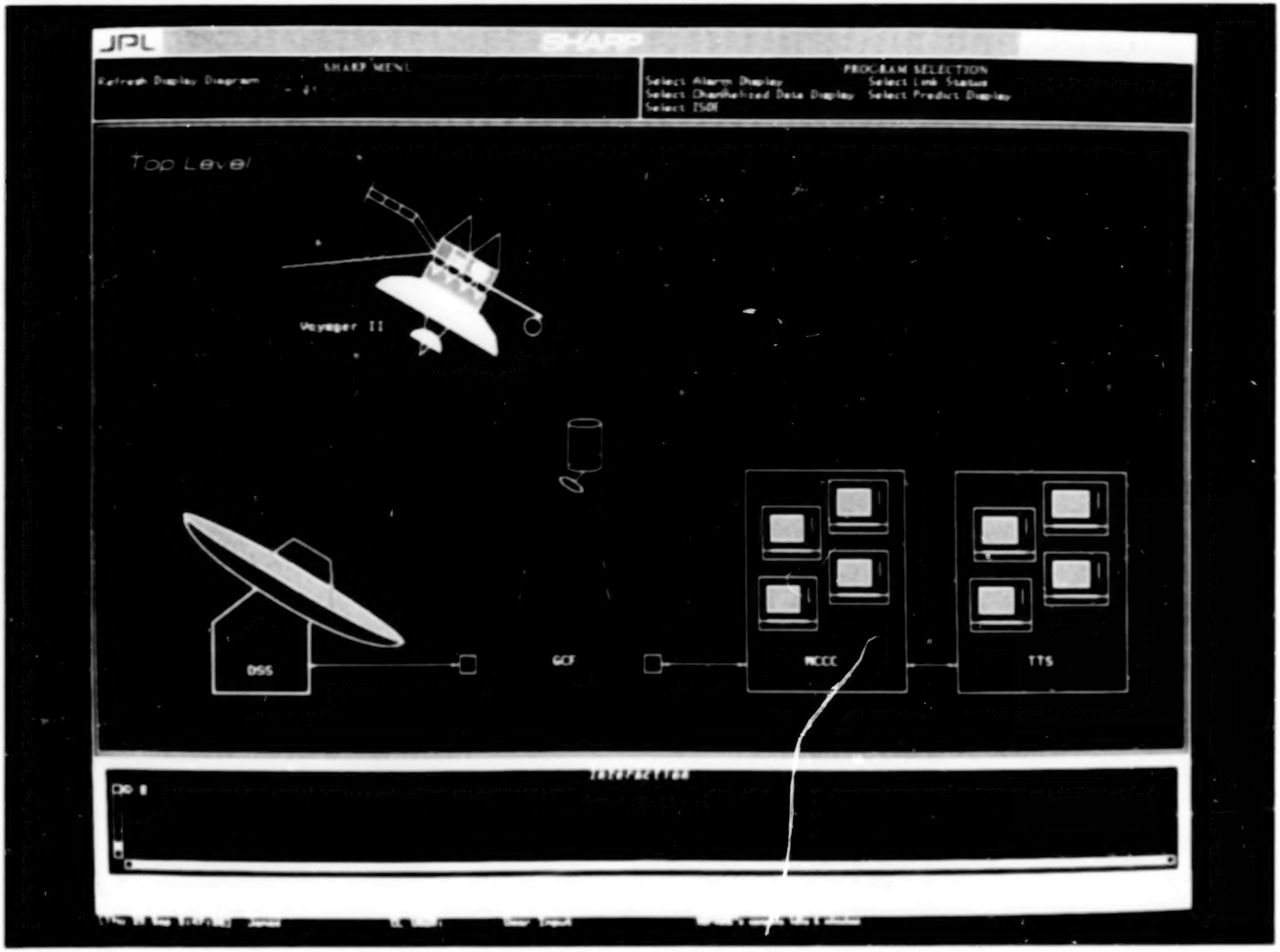

JPL
SHARP
SHARP MENU
Refresh Display Diagram
PROGRAM SELECTION
Select Alarm Display
Select Link Status
Select Channelized Data Display
Select Predict Display
Select ISOE
Top Level
Voyager II
DSS
GCF
MCCC
TTS
INTERACTION

Figure 2. SHARP Top Level System Status View.

BEST COPY AVAILABLE

x-band modulation index is 32, the two drivers are on, the subcarrier frequency is high, and the data line rate is high. Translation of these spacecraft commands from their raw form into more understandable summaries of spacecraft activity may be performed, and the user can request status summaries of any activity. A history display is maintained as the ISOE is updated so that the user can verify modifications.

SHARP's display which plots the channelized data, illustrated in Figure 3, is a significant improvement over existing capabilities. The user dynamically customizes the display at any time by selecting which and how many channels to view, the time scale, the data range for each plot, and even the icon to use for graphing points on each channel. Each plot is color-coded by the user for easy visual distinction between displayed channels. When any channel is in alarm, its corresponding data points are plotted in red, facilitating rapid detection of an alarm condition. The channel's associated alarm limits may be optionally overlaid onto the channel's plot for further information. Each data point is mouse sensitive to provide time and numerical value indicators, and an automatic counter continually indicates the number of data points per plot. Pan and zoom features augment this display, which can represent information as graphs of actual or derived data vs. time, xy plots, scatter plots, or logarithmic scales.

The SHARP system also provides an alarm limit interface which allows on-line viewing and editing of established spacecraft engineering alarm limits, DSS performance limits, ground data system limits, and residual thresholds. Authorized users may permanently alter any of these limits, and specified values may be changed temporarily for the remainder of that particular spacecraft pass. The later capability, manual override, enables alarm suppression or closer scrutiny for any particular event with no intervening paperwork.

Among the new graphical analysis capabilities provided by the SHARP system is the Telecom link status display, as shown in Figure 4. Actual station coverage is illustrated, along with spacecraft transmitter power status, data rate, data outages, and real-time recording of station uplink (signal transmission) and projected downlink a round trip light time later. Detailed analysis is performed and information is subsequently color-coded to represent changes in status. The display provides the user with such valuable information as time ranges and explanations of data outages (i.e. no station coverage or ongoing spacecraft maneuver), and can warn the operator when to expect noisy data and why.

The SHARP system also features on-line functional block diagram schematics of the end-to-end communications path from the spacecraft through the Deep Space Communications Complex and Ground Communications Facility to the Mission Control Center at JPL and final destination of the Test and Telemetry System computers. Each top level system status may be viewed at successive levels of detail. The Telecom subsystem is very comprehensive, as spacecraft schematics have been developed for the all of its individual components. These dynamic block diagrams are driven by various ISOE status indicators and the channelized data. The status of spacecraft and DSS components (operational, off-line, or in alarm) is depicted by color, facilitating rapid status identification at a glance. Figure 5 shows the Telecommunications subsystem and Figure 6 illustrates the spacecraft receiver with associated diagnostic messages.

Another graphical display which combines various sources of information and data is SHARP's Attitude and Articulation Control display. This display

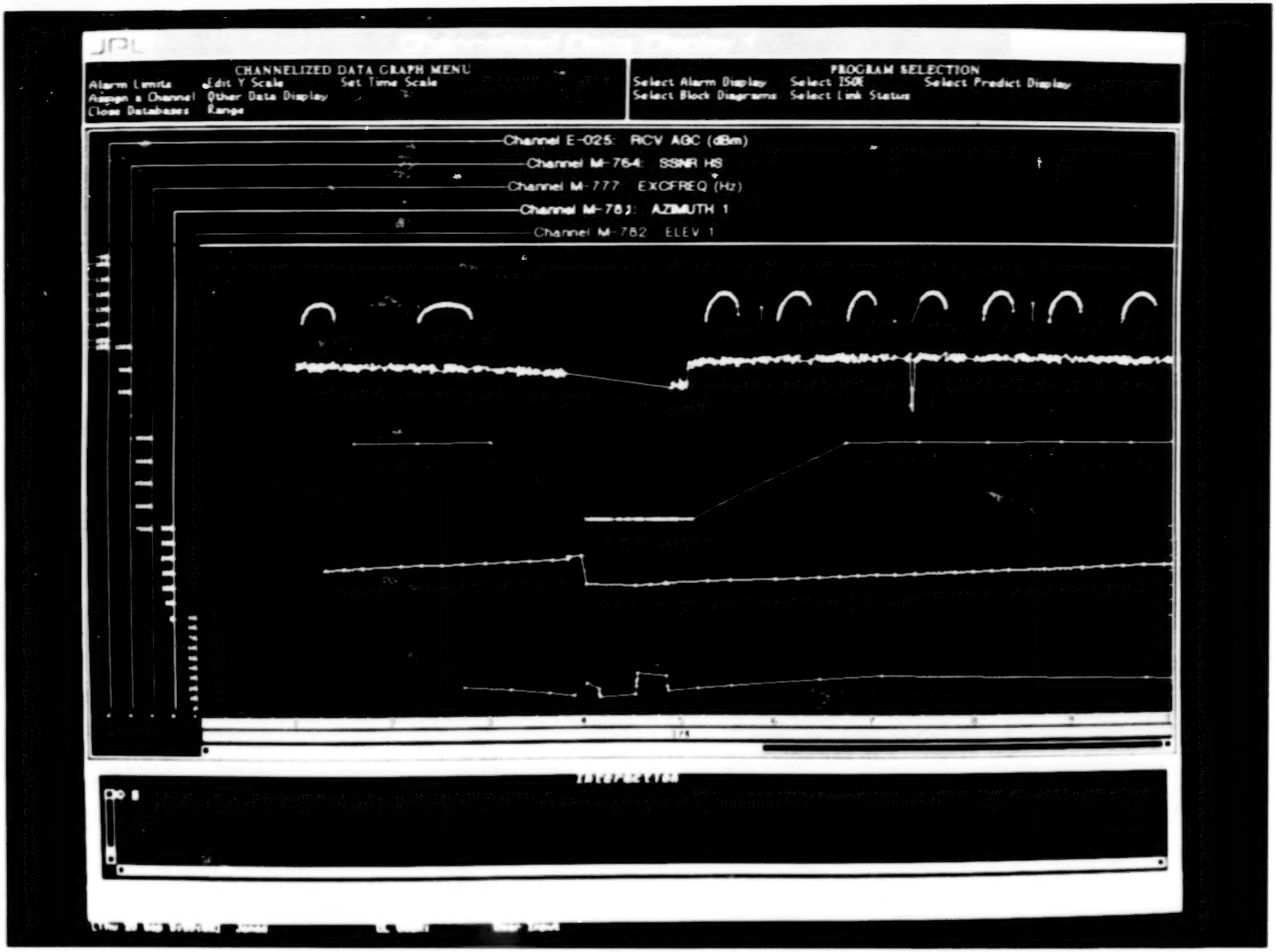

JPL
CHANNELIZED DATA GRAPH MENU
Alarm Limits
Edit Y Scale
Set Time Scale
Assign a Channel
Other Data Display
Close Databases
Range
PROGRAM SELECTION
Select Alarm Display
Select ISOE
Select Predict Display
Select Block Diagrams
Select Link Status
Channel E-025: RCV AGC (dBm)
Channel M-764: SSNR HS
Channel M-777: EXCFREQ (Hz)
Channel M-781: AZIMUTH 1
Channel M-782: ELEV 1

Figure 3. Plotting capabilities available in SHARP.

BEST COPY AVAILABLE

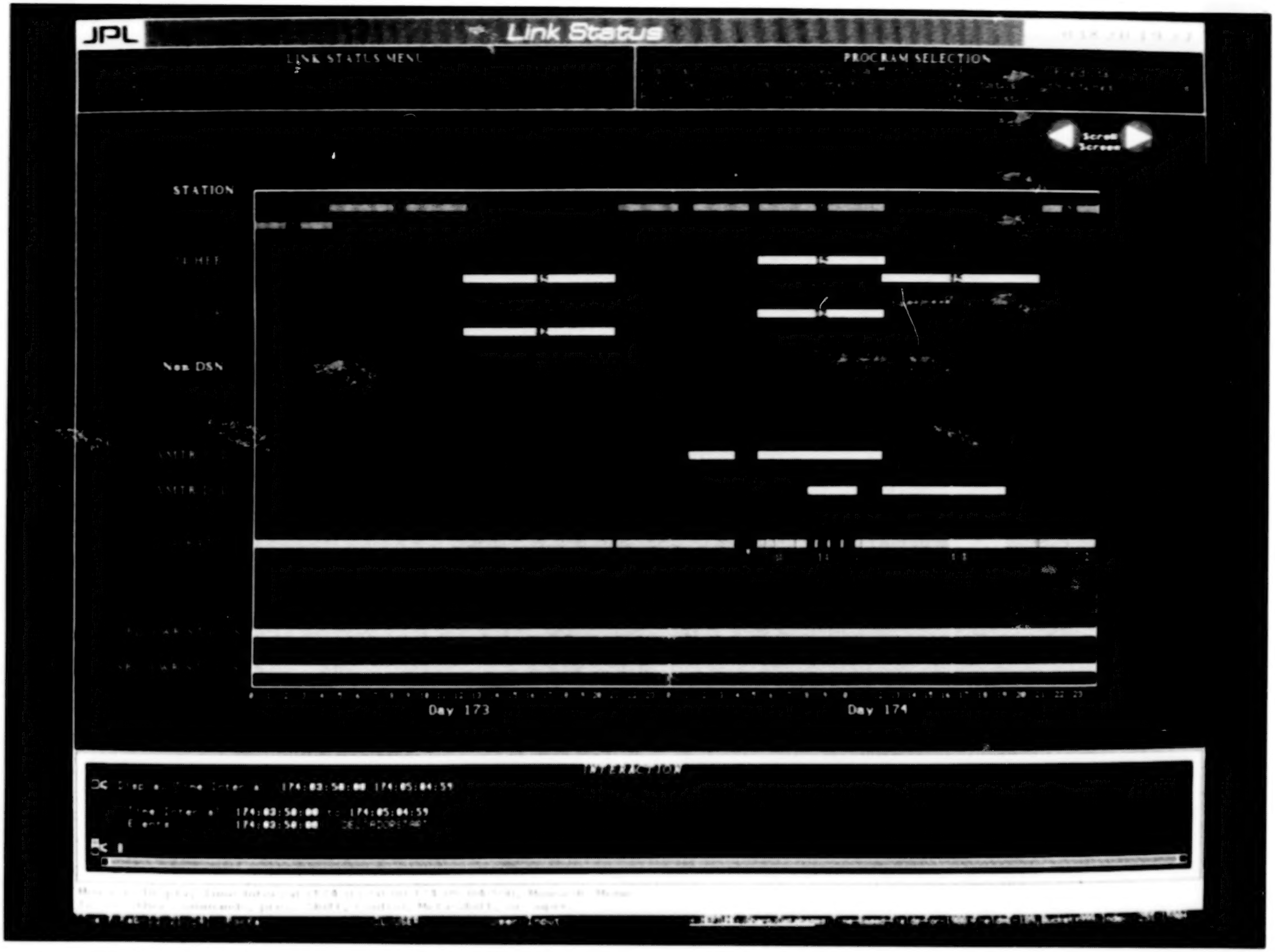

JPL
Link Status
LINK STATUS MENU
PROGRAM SELECTION
Scroll Screen
STATION
Non DSN
Day 173
Day 174
INTERACTION

Figure 4. SHARP Telecommunications Link Status display.

BEST COPY AVAILABLE

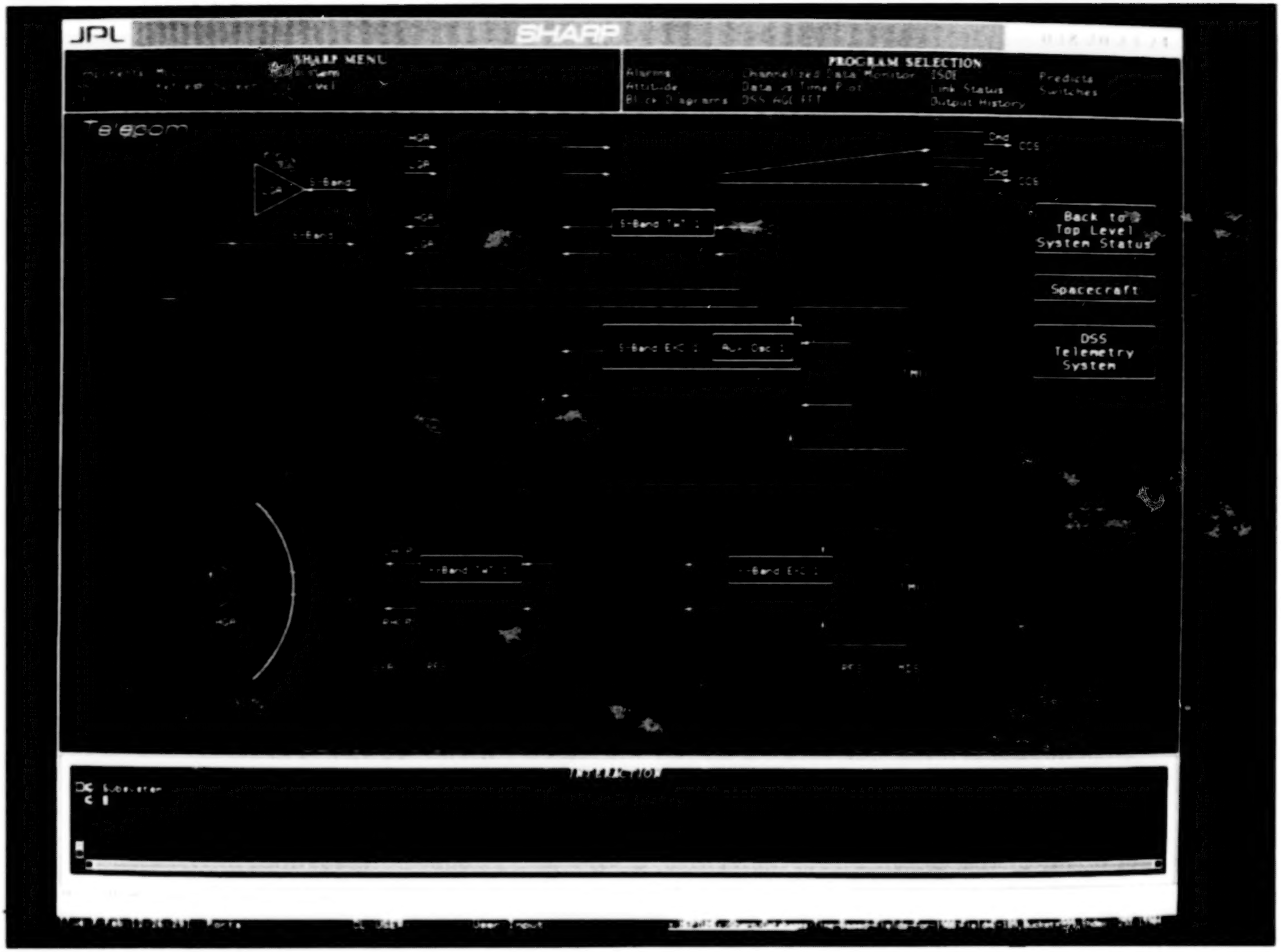
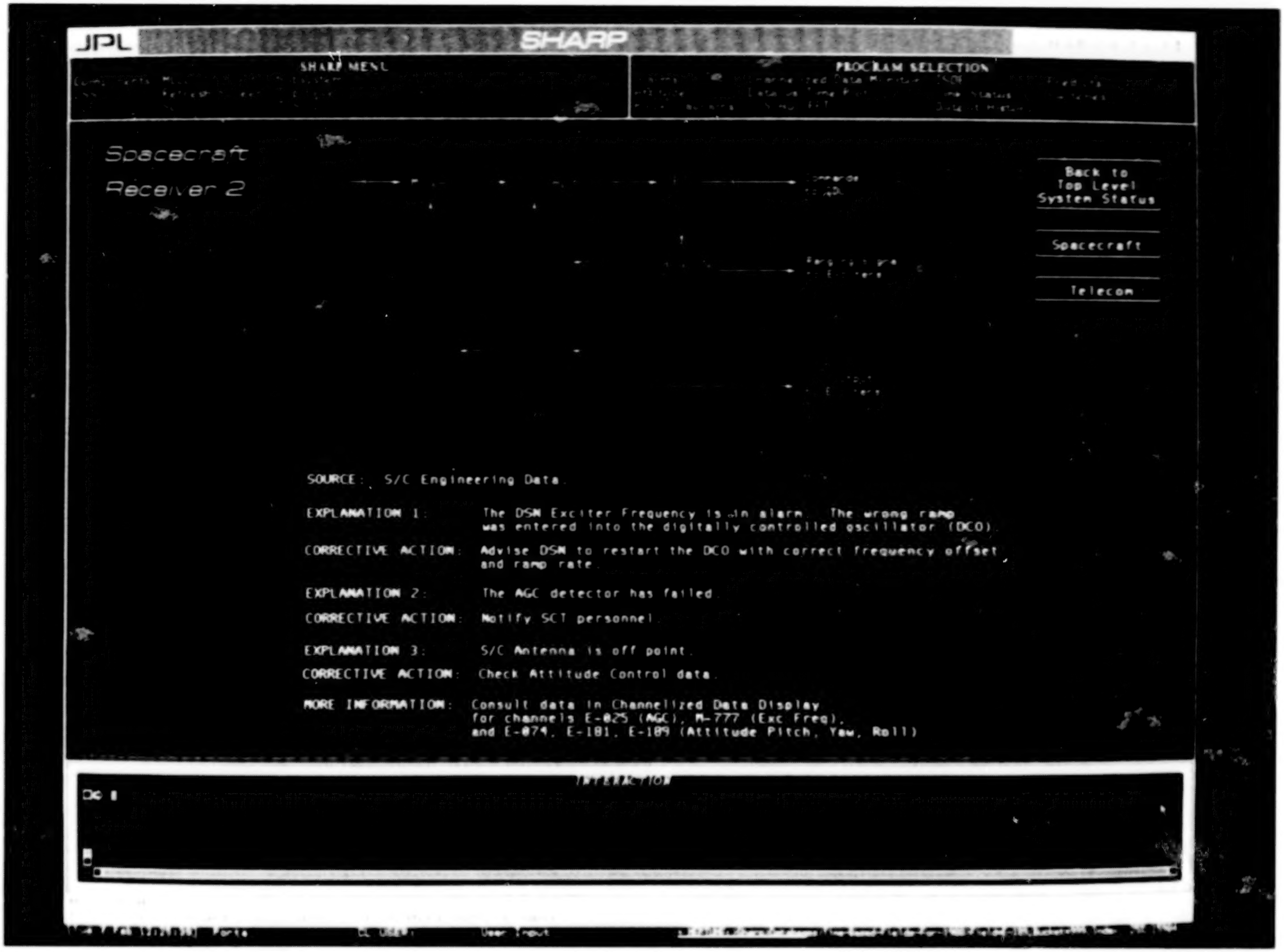

JPL
SHARP
SHARP MENU
PROGRAM SELECTION
Alarms
Channelized Data Monitor
ISOE
Predicts
Attitude
Data vs Time Plot
Link Status
Switches
Block Diagrams
DSS AGC FFT
Output History
Telecom
HGA
LGA
S-Band
X-Band
S-Band TWT 1
S-Band EXC 1
Aux Osc 1
X-Band TWT 1
X-Band EXC 1
Cmd
CCS
Back to Top Level System Status
Spacecraft
DSS Telemetry System
INTERACTION

Figure 5. SHARP schematic block diagram of Voyager Telecommunications subsystem.

BEST COPY AVAILABLE


JPL
SHARP
SHARP MENU
PROGRAM SELECTION
Spacecraft
Receiver 2
Back to
Top Level
System Status
Spacecraft
Telecom
INTERACTION

SOURCE: S/C Engineering Data.

EXPLANATION 1: The DSN Exciter Frequency is in alarm. The wrong ramp was entered into the digitally controlled oscillator (DCO).

CORRECTIVE ACTION: Advise DSN to restart the DCO with correct frequency offset and ramp rate.

EXPLANATION 2: The AGC detector has failed.

CORRECTIVE ACTION: Notify SCT personnel.

EXPLANATION 3: S/C Antenna is off point.

CORRECTIVE ACTION: Check Attitude Control data.

MORE INFORMATION: Consult data in Channelized Data Display for channels E-025 (AGC), M-777 (Exc Freq), and E-074, E-181, E-189 (Attitude Pitch, Yaw, Roll)

Figure 6. SHARP schematic block diagram of spacecraft receiver with message from diagnostician.

BEST COPY AVAILABLE

combines spacecraft motion parameters (pitch, yaw, and roll) and projects spacecraft movement over time. A limit cycle box which represents defined spacecraft deadband limits encloses the spacecraft icon. Alarm conditions are easily detected as the spacecraft icon drifts outside of the designated deadband box. Trailing vectors establish the path of the spacecraft.

The SHARP system also contains special processing modules to perform subsystem specific analysis. For the Telecommunications subsystem, a Fast Transform (FFT) of the DSS conical scanning component is performed to indicate when the antenna is going off point. This is a relatively common event, which currently may take hours to detect and correct. Spacecraft and scientific information can be permanently lost when this situation occurs. SHARP's FFT display illustrates the results of a Fast Fourier Transform process performed on 64 data points of a particular channel, and provides instant information on conical scan error. The problem can be detected in a matter of minutes, and the station can be contacted to correct the antenna movement prior to the loss of data.

Artificial Intelligence

The artificial intelligence modules of SHARP are written in an expert system building language called STAR*TOOL. STAR*TOOL is a programming language designed at JPL to meet many of NASA's demanding and rigorous AI goals for current and future projects. Appendix A contains a more detailed description of the STAR*TOOL system.

Artificial intelligence techniques are distributed throughout all components of the SHARP system. Intelligent programming methodologies such as heuristic adaptive parsing, truth maintenance, and expert system technology enable more effective automation and thorough analysis for SHARP functions. Fault detection becomes almost immediate with a greater degree of accuracy and precision, and the system quickly generates fault hypotheses.

A blackboard architecture, provided by STAR*TOOL, serves as a uniform framework for communication within the heterogeneous multi-process environment in which SHARP operates. Generally, when two or more processes are cooperating, they must interact in a more complicated manner than simply setting global variables and passing information along such paths. SHARP provides a standardized method of communication between multiple processes, which include real-time posting of incoming telemetry data and the monitoring of data networks.

Heuristic adaptive parsing is implemented for SHARP's raw predicts database. Periodically the format of this data source changes without mission operations being notified. Generally this would require the raw predicts parser to be rewritten to incorporate the new format. However, SHARP utilizes Augmented Transition Network [2] (ATN) techniques to accomplish adaptive parsing. The advantage of such an ATN lies in its ability to parse the database according to semantic content rather than syntactic structure. The raw predicts database can therefore be modified and yet remain successfully parsable. This heuristically controlled format insensitive parsing ensures continuity despite format modifications in predict generation.

The centralized database of the SHARP system serves as a central repository of all real-time and non-real-time data, and functions as a local buffer to enable rapid data access for real-time processing. Numerous database manipulation functions have been implemented, and database daemons have been constructed to implement spontaneous computations. Requests can be made to the database to trigger arbitrary activities when a complex combination of past, present, and future events occur. A wide selection of retrieval methods by time or value highlight the flexibility inherent in the database. Requests to the database can be made from both AI and non-AI modules of SHARP, and can be handled serially or in parallel.

Various SHARP modules represent and manipulate data symbolically rather than numerically so that particular numeric values can change without forcing the algorithms themselves to be modified. For example, to determine if a channel is in alarm, the rule interpreter manipulates one symbolic fact, ChannelInAlarm, rather than the many numeric operations that are required to make an actual determination. This is a significant advantage as SHARP presently analyzes over 100 channels, and the alarm determination process varies from channel to channel. Symbolic representation and manipulation of data also simplifies the exchange of information between SHARP modules and reduces reliance on specific dimensionless numeric values.

The diagnostic component of SHARP is composed of a hierarchical executive diagnostician coupled with cooperating and non-cooperating mini-experts. Each mini-expert is responsible for the local diagnosis of a specific fault or class of faults, such as particular channels in alarm, conical scan errors, or loss of telemetry. A non-cooperating expert focuses only on its designated fault area, but a cooperating expert has the additional capability of searching beyond its local area to identify related faults that are likely to occur. Cooperating experts are used in situations where the identification of a particular fault cannot be made by examining a single fault class alone.

The executive diagnostician combines input propagated from each local diagnostician and reviews the overall situation to propose one or more fault hypotheses and recommended corrective actions. When multiple fault hypotheses are generated, the system lists all possible causes of the anomaly and ranks each according to plausibility.

If one or more of the cooperating experts fails, the executive diagnostician will continue to operate with only a reduction in the area of local diagnosis that would have been derived from the failed mini-experts. Similarly, if the executive diagnostician fails, the cooperating experts will locally diagnose the faults in isolation of multiple fault consideration.

The diagnostician is implemented in rules that execute in pseudo-parallel in pursuit of multiple hypotheses. Pseudo-parallelism is implemented in SHARP using facilities provided by STAR*TOOL, which includes parallelism as a fundamental control structure. The diagnostic rules operate in isolation of one another by executing in independent contexts provided by the STAR*TOOL memory model, and communicate through the Blackboard facility.

These contexts can be organized into a tree-like structure to represent contradictory information resulting from changes in facts or from the introduction of new or contradictory hypotheses. Facilities in the truth maintenance system handle data and demand driven diagnoses to ensure an appropriate balance between the persistence of hypotheses and sensitivity to new data.

Bayesian inference processes are used for comparing multiple hypotheses and for prioritizing conflicting fault hypotheses. Bayesian inference procedures also perform uncertainty management to allow continued high performance in the presence of noisy, faulted, or missing data.

The truth maintenance system constantly monitors for violations of logical consistency. For example, it performs conflict checking to maintain consistency among multiple rule firings, hypotheses, and the knowledge base, and allows the context sensitive management of alarms through a complex response system to combinations of alarm conditions. Truth maintenance techniques also provide a variety of functions for temporal reasoning in multiple fault diagnosis.

CONCLUSIONS

Spacecraft and ground data systems operations present a rigorous environment in the area of monitoring and anomaly detection and diagnosis. With a number of planetary missions scheduled for the near future, the effort to staff and support these operations will present significant challenges.

The SHARP system is an attempt to address the challenges of a multi-mission monitoring and troubleshooting environment by augmenting conventional automation technologies with state-of-the-art artificial intelligence. Results of this effort to date have already shown significant improvements over current Voyager methodologies and have provided enhancements to several aspects of Voyager operations.

This type of automation technology will endow mission operations with considerable benefits. In as many areas as are automated, expert knowledge will be captured and permanently recorded, reducing the frenzied state that occurs when domain specialists announce their impending retirement. Cost reductions will occur as a result of automation and decreased requirement for 24-hour real-time operator coverage. Automatic fault detection and analysis will facilitate quicker response times to mission anomalies and more accurate conclusions. The time savings afforded by SHARP-like capabilities, especially during periods of unmanned operation or during emergencies, could mean the difference between the loss or retention of critical data, or possibly even the spacecraft itself.

ACKNOWLEDGEMENTS

The authors wish to acknowledge the following people for their participation in the SHARP project: David J. Atkinson, task management; Harry J. Porta and Gaius Martin, software development; Boyd Madsen, expert knowledge for Voyager and Galileo spacecraft telecommunications; and Bruce Elgin and Erann Gat, software contributions.

The work described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

REFERENCES

[1] James, Mark L., *STAR*TOOL Reference Manual*, Internal Document, Jet Propulsion Laboratory, Pasadena, California (1988).

[2] Charniak, Eugene and McDermott, Drew, *Introduction to Artificial Intelligence*, Addison-Wesley (1985).

APPENDIX A: STAR*TOOL

Knowledge-based systems for automated task planning, monitoring, diagnosis, and other applications require a variety of software modules based on artificial intelligence concepts and advanced programming techniques. The design and implementation of such modules requires considerable programming talent and time, and a background in theoretical artificial intelligence. Sophisticated software development tools that can speed the research and development of new artificial intelligence applications are therefore highly desirable. The STAR*TOOL system, designed and built by

Mark James, was developed specifically for this purpose. Included in the system are facilities for developing reasoning processes, memory-data structures and knowledge bases, blackboard systems, and spontaneous computation daemons.

Computational efficiency and high performance are especially critical in artificial intelligence software. This consideration has been an important objective of STAR*TOOL, and has led to its design as a toolbox of AI facilities that may be used independently or collectively in the development of knowledge-based systems.

STAR*TOOL provides a variety of facilities for the development of software modules in knowledge-based reasoning engines. The STAR*TOOL system may be used to develop artificial intelligence applications as well as specialized tools for research efforts.

STAR*TOOL facilities are invoked directly by the programmer in the Common LISP language. For improved efficiency, an optional optimization compiler was developed to generate highly optimized CommonLISP code.

STAR*TOOL was designed to be efficient enough to operate in a real-time environment and to be utilized by non-LISP applications written in conventional programming languages such as ADA, C, Fortran, and Pascal. These non-LISP applications can run in a distributed computing environment on remote computers, or on a computer that supports multiple programming languages.

REDEX: The Ranging Equipment Diagnostic Expert System

Edward C. Luczak and K. Gopalakrishnan
Computer Sciences Corporation
4600 Powder Mill Rd.
Beltsville, MD 20705

David J. Zillig
Telecommunication Systems Branch, Code 531
NASA Goddard Space Flight Center
Greenbelt, MD 20771

ABSTRACT

This paper describes REDEX, an advanced prototype expert system that diagnoses hardware failures in the Ranging Equipment (RE) at NASA's Ground Network tracking stations. REDEX will help the RE technician identify faulty circuit cards or modules that must be replaced, and thereby reduce troubleshooting time. It features a highly graphical user interface that uses color block diagrams and layout diagrams to illustrate the location of a fault.

A semantic network knowledge representation technique was used to model the design structure of the RE. A catalog of generic troubleshooting rules was compiled to represent heuristics that are applied in diagnosing electronic equipment. Specific troubleshooting rules were identified to represent additional diagnostic knowledge that is unique to the RE. Over 50 generic and 250 specific troubleshooting rules have been derived.

REDEX is implemented in Prolog on an IBM PC AT-compatible workstation. Block diagram graphics displays are color-coded to identify signals that have been monitored or inferred to have nominal values, signals that are out of tolerance, and circuit cards and functions that are diagnosed as faulty. A hypertext-like scheme is used to allow the user to easily navigate through the space of diagrams and tables. Over 50 graphic and tabular displays have been implemented.

REDEX is currently being evaluated in a stand-alone mode using simulated RE fault scenarios. It will soon be interfaced to the RE and tested in an online environment. When completed and fielded, REDEX will be a concrete example of the application of expert systems technology to the problem of improving performance and reducing the lifecycle costs of operating NASA's communications networks in the 1990's.

1.0 INTRODUCTION

One of NASA's big challenges for the 1990's is to contain the operations and maintenance costs of its mission operations and ground support systems while at the same time sustaining the current high levels of mission performance and readiness. Expert systems technology is being evaluated by several NASA organizations because of its potential for capturing the specialized expertise of operations and maintenance personnel and making it available rapidly, consistently, around the clock, and at multiple locations.

One of the more promising areas for applying expert systems technology is in the area of problem and fault diagnosis. Prototype "diagnostic assistant" tools are being developed to help operators and engineers detect, isolate, diagnose, and in some cases recover from operational problems and system faults.

A number of prototype expert systems are being developed and evaluated by NASA to diagnose problems onboard orbiting spacecraft, including the Tracking and Data Relay Satellite (TDRS) [1], the Hubble Space Telescope [2], and the Gamma Ray Observatory (GRO) [3]. Another prototype is being developed to diagnose faults in communications links with the Cosmic Background Explorer (COBE) [4].

Expert systems are also being evaluated for use in detecting and diagnosing faults in NASA ground systems. Use of diagnostic assistant tools can reduce the amount of time required to diagnose complex ground systems, and allow them to be repaired and restored to mission support more rapidly. Diagnostic expert systems for NASA ground systems are being developed and evaluated for Shuttle launch preparation hardware [5], telemetry processing systems [6], and an institutional local area network (LAN) [7]. An advanced fault isolation expert system that diagnoses problems in the ground and space components of the NASA Space Network [8,9] is approaching operational use.

This paper is a case study description of the Ranging Equipment Diagnostic Expert System (REDEX). REDEX is an advanced prototype expert system that diagnoses hardware failures in the Ranging Equipment (RE) at NASA's Ground Network tracking stations. When an RE failure occurs, REDEX will help the RE technician identify the faulty circuit cards or modules that must be replaced. REDEX is designed to assist the technician by significantly reducing troubleshooting time. It features a highly graphical user interface that uses color block diagrams and layout diagrams to illustrate the location of a fault. When testing and evaluation are completed, REDEX is planned to be installed at several NASA tracking stations.

The following sections describe the characteristics of the RE diagnosis problem (Section 2), the approach used in developing REDEX (Section 3), key aspects of the design of the expert system (Section 4), the current status of the implementation (Section 5), key observations and conclusions that resulted from this work (Section 6), and directions for future work (Section 7).

2.0 THE PROBLEM

The NASA Ground Network (GN) consists of a set of tracking stations that are used to track Space Shuttle launches and various unmanned spacecraft. The GN and its affiliated facilities include tracking systems at Bermuda, Merritt Island, FL, and Wallops Island, VA, and engineering test beds at Goddard Space Flight Center (GSFC) and Johnson Space Center (JSC).

The RE [10] is one of the principal equipment items in the S-Band ranging system at the tracking stations (Figure 1). The ranging system uses a 9-meter dish antenna to send and receive tracking signals from the target spacecraft. The RE determines the range and Doppler (velocity) of the spacecraft, and transfers this data to a tracking data processor. The data is then transmitted via the NASA Communications Network (NASCOM) to GSFC.

The RE occupies one full equipment rack, and includes over 50 custom-designed circuit boards, seven power supplies, and an embedded PDP-11/73 computer and color display terminal. The equipment includes a variety of high and low frequency analog circuitry, digital electronics, and several embedded microprocessors. It is arguably the most diverse and complex single rack of electronics at a GN tracking station. When a hardware failure occurs in the RE, the technician must identify the faulty circuit board or module and replace it with an on-site spare as rapidly as possible to restore the RE to service for mission support.

The RE design includes a Ranging Internal Monitor (RIM) system consisting of 74 test points that are distributed throughout the RE. These RIM points continuously monitor key signal strengths, DC voltages, frequencies, and logic levels. The RIM points status is gathered and reported to the PDP-11 computer, which displays the data upon request to the RE operator. RIM points that have strayed from their nominal range are displayed as red, and those within tolerance are shown in green. A fault is indicated when one or more of the RIM points are red. The RE technician uses the array of RIM point status data to begin troubleshooting the failure.

The troubleshooting process may take an hour or more, requiring the technician to trace signals and status through a set of several hundred block diagrams and circuit schematics. A major difficulty is that a single fault often causes many RIM points on different circuit boards to become red, because of the propagation of bad signals throughout the RE. The diagnostic troubleshooting process involves reasoning about signal flow and cause-effect relationships in electronic systems. This reasoning process requires knowledge of the available RIM point values, knowledge of general electronic troubleshooting rules, and specific knowledge of the design and operation of the RE.

The REDEX diagnostic assistant was conceived to reduce RE troubleshooting time, which is especially critical in the event of an RE failure during a Shuttle launch countdown. This problem is appropriate for an expert system solution because it involves reasoning processes that can be well defined, and requires specific knowledge spanning only a narrow domain. As shown in Figure 1, REDEX resides on a workstation and obtains the RIM point status data from the RE via a communications link. When the RE internal monitoring system has detected a fault, REDEX can diagnose the situation and identify the circuit cards that are suspected to be faulty. REDEX can display the RIM point status data and the results of the diagnosis using displays of RE block diagrams, rack and drawer layout diagrams, and tabular data.

3.0 APPROACH

Compared to traditional software development efforts, the software industry has relatively little experience in developing expert systems. It is widely recognized that the successful development of expert systems requires a methodology that differs from those typically used for

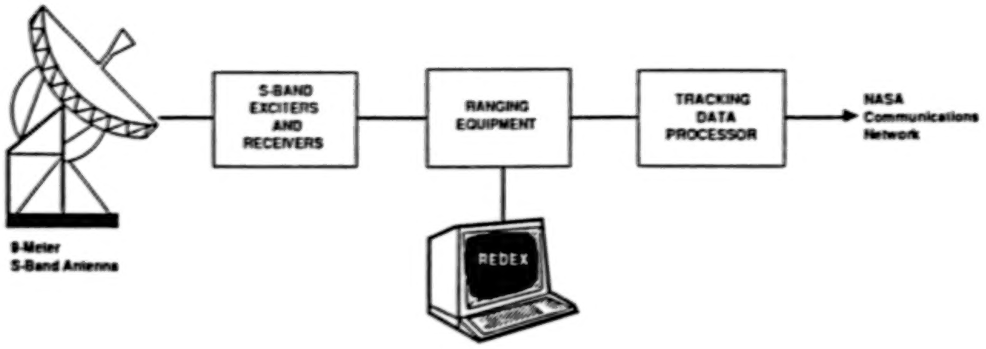


Figure 1. NASA Ground Network Tracking Station

traditional software. Unfortunately, insufficient data has been gathered to allow the evaluation and selection of such a methodology with confidence.

REDEX is being developed using a methodology that has several of the characteristics discussed in [11]. The approach has followed the overall framework of the standard software lifecycle "waterfall model", involving requirements definition, design, implementation, and testing. Requirements were defined in a knowledge acquisition and knowledge representation process, and then formalized in a requirements document [12]. The software design was developed and specified in a design document [13].

However, iteration is a key component of the REDEX development approach, and has been driven by a continuing evolutionary prototyping effort. (See Section 3.3.) The requirements were first informally defined, then prototyped, and then formally documented. The design followed the same sequence. This approach allowed the requirements and the design to be evaluated and validated via prototyping prior to formal documentation. Both the requirements definition and design were allowed to evolve during the prototyping. This iterative approach is characteristic of the "spiral model" of software development. REDEX is being developed using many elements of the spiral-based prototyping methodology described in [14] and [15].

The documentation steps were valuable because they provided (a) the discipline of clearly defined development milestones, (b) a mechanism for careful definition of the state of the requirements and design, and (c) a vehicle for external review and validation. Both the requirements and design documents will be periodically updated.

The following sections describe several aspects of the approach in more detail.

3.1 KNOWLEDGE ACQUISITION

The knowledge needed to develop REDEX was acquired from the three sources described below. The knowledge acquisition team included persons familiar with electronic systems, fault analysis, and knowledge representation.

(a) Technical documentation - An intensive study of RE design documentation, circuit diagrams, operations manuals, and maintenance documentation was conducted.

(b) Training class - REDEX knowledge engineers attended the NASA training class that teaches new operators and technicians how to operate, troubleshoot, and repair the RE. The class included hands-on operation of the RE.

(c) Interviews with expert - The principal instructor for the training class is an expert in troubleshooting the RE; he was interviewed to obtain additional detailed information.

3.2 KNOWLEDGE REPRESENTATION

The knowledge required to diagnose RE faults includes knowledge of the design of the RE and knowledge of the procedures or rules needed for troubleshooting. The techniques described below were used to represent this knowledge, and to document it in [12].

Knowledge of the RE design was represented using semantic networks. A model of the RE design structure was constructed representing RE design objects and their relationships. The objects represented include signals, RIM test points, circuit boards, functions, switches, power supply voltages, circuit drawers, and functional subsystems. A function is generally a portion of a circuit board.

Figure 2 illustrates the full set of object classes used in the RE model, and the relationships that can exist between instances of these classes. Both hierarchical and heterarchical relationships are included. An object instance is characterized by a set of values for attributes that are associated with its class.

Semantic network diagrams were drawn to capture the relationships "input-to", "output-from", "supplies", and "status-of" between signals, RIM points, power expressions, and functions. Diagramming conventions were defined to simplify the construction and interpretation of these diagrams. They have the appearance of structured block diagrams. Approximately 20 semantic network diagrams were constructed and included in [12]. They provide a readable, formal definition of the lowest level of the RE model that REDEX needs for diagnostic reasoning.

Troubleshooting knowledge was represented using rules in an "IF (conditions) THEN (conclusions)" format. A notation for representing these rules was developed that is independent of any specific implementation language. This independence is analogous to the way that Program Design Language (PDL) used in traditional software design is independent of the language to be used for coding. A simple example rule is:

IF red(RIM22) and green(RIM29)
and green(RIM31)
THEN suspect(a6a13f1)

These rules are called "specific troubleshooting rules" because they represent specific knowledge about assessing the health of a function or a signal in the RE. Over 250 specific rules were derived.

The specific rules were derived from the semantic network diagrams by searching for certain patterns of monitored and unmonitored signals around function boxes that enable diagnostic inferences to be made. These patterns are called "generic troubleshooting rules". Figure 3 illustrates two simple generic rules. A catalog of over 50 generic rules was compiled. These generic rules represent general troubleshooting knowledge and are applicable to other equipment that is similar to the RE.

3.3 PROTOTYPING

REDEX has been developed using an evolutionary prototyping process, in which a series of five prototype stages were defined and built. This process could be viewed as a spiral, in which each stage involves definition, implementation, testing, and evaluation activities. Each prototype stage was only a few months in duration. Each stage was defined such that its successful completion would reduce development risks in an important area.

The first stage began early in the requirements definition activity; its objective was to demonstrate the feasibility of representing RE troubleshooting rules in the Prolog language. A small set of rules required to diagnose faults in one functional subsystem of the RE was implemented and demonstrated as a "proof-of-concept".

Successful completion of this first stage led to the second prototype stage, whose objective was to encode the knowledge needed to diagnose faults in the entire RE. This prototype was implemented incrementally, adding and testing the knowledge for one RE subsystem at a time until all ten subsystems were implemented. The requirements document was written when this stage was successfully completed and demonstrated.

The objective of the third prototype stage was to develop and evaluate the concepts of the user interface design. (The first two prototype stages involved only a primitive user interface.) Development and demonstration of this prototype led to many improvements in the design concept. The objective of the fourth prototype stage was to fully implement the user interface. The design document was written when this stage was successfully completed and demonstrated.

The fifth prototype stage is currently underway. In this stage, REDEX is being refined to correct diagnostic errors discovered during testing, made more robust with error handling capabilities, and enhanced to incorporate feedback received from user evaluations. This stage also adds the capability to communicate with the RE.

4.0 DESIGN

The design of REDEX defines a system environment, a control structure for implementing the knowledge described in Section 3.2, a user interface, and a data interface with the RE. The design evolved and was tested using the prototyping process discussed in Section 3.3. The following sections describe several key aspects of the design.

4.1 SYSTEM DESIGN

REDEX is designed to execute on an IBM PC AT-compatible workstation with a high resolution (640x480 pixels) color display. The software is written in the Turbo Prolog language, which runs under MS-DOS. An evaluation of the effectiveness of this language for implementing REDEX can be found in [16]. Turbo Prolog was selected for REDEX implementation because it provides: (a) a straightforward and efficient representation of the troubleshooting rules, (b) the facilities and flexibility needed for building the graphical interface, (c) a good software development environment, (d) excellent runtime

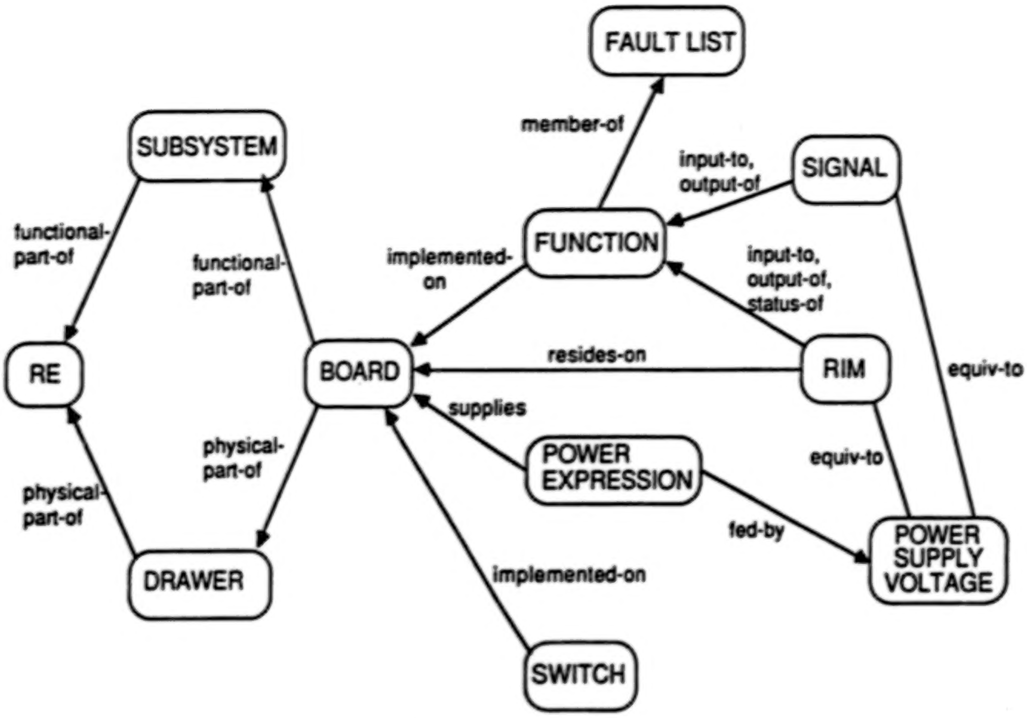

FAULT LIST
member-of
SUBSYSTEM
input-to, output-of
SIGNAL
functional-part-of
functional-part-of
implemented-on
FUNCTION
input-to, output-of, status-of
RE
BOARD
resides-on
RIM
equiv-to
supplies
physical-part-of
POWER EXPRESSION
equiv-to
physical-part-of
fed-by
POWER SUPPLY VOLTAGE
DRAWER
implemented-on
SWITCH

Figure 2. REDEX Object Classes and their Relationships

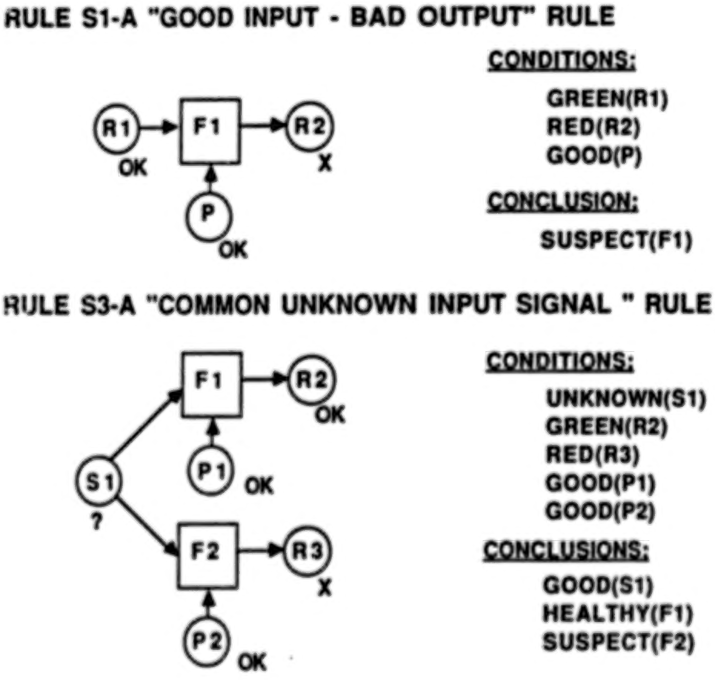

RULE S1-A "GOOD INPUT - BAD OUTPUT" RULE
CONDITIONS:
GREEN(R1)
RED(R2)
GOOD(P)
CONCLUSION:
SUSPECT(F1)
R1
OK
F1
R2
X
P
OK
RULE S3-A "COMMON UNKNOWN INPUT SIGNAL " RULE #1
CONDITIONS:
UNKNOWN(S1)
GREEN(R2)
RED(R3)
GOOD(P1)
GOOD(P2)
CONCLUSIONS:
GOOD(S1)
HEALTHY(F1)
SUSPECT(F2)
F1
R2
OK
P1
OK
S1
?
F2
R3
X
P2
OK

Figure 3. Example Generic Troubleshooting Rules

performance, and (e) it is significantly less costly than other expert system development tools..

Figure 4 shows the REDEX system block diagram. The knowledge base contains Prolog clauses that represent facts and rules. The inference engine is provided by the built-in control structure of Prolog. Procedures are implemented using Prolog predicates.

4.2 DIAGNOSTIC FUNCTIONS

REDEX uses a rule-based design approach to implement the RE troubleshooting rules. The design also incorporates a representation of the objects in the RE and their relationships. The overall control structure is goal-directed backward rule chaining. Figure 5 is an RE fault tree that illustrates the goal decomposition. The top node, "Diagnosis", represents the primary goal of determining the reason for an RE red health status (i.e. one or more RIM points are red). The second level of nodes represents three types of possible reasons: that an RE input signal is bad, that an RE function contains a fault, and that a reportable status condition that affects RE health status has occurred. The top level "Diagnosis" goal is satisfied if one of the second level goals is satisfied.

The third level nodes in Figure 5 represent specific reasons for red health status, e.g. specific functions that could contain a fault. The fourth level nodes are troubleshooting rules (R). When one of these rules fires, its conclusion may be that a signal is good or bad with a given confidence factor, that a function or set of functions is suspected to contain a fault, or that a reportable status condition exists. The conditions in these rules involve the color of RIMs, the positions of RE configuration switches, the values of power expressions, and the status of evaluated signals. In order to evaluate these conditions, many lower-level rules not shown in the figure may have to be fired.

Although it is unlikely that more than one fault will occur at a time, REDEX always searches the entire fault tree in order to detect multiple fault conditions if they occur.

4.3 USER INTERFACE

The REDEX user interface design is based on the use of menus for soliciting user command entries, and the use of color-coded graphic diagrams and tables for displaying RE status data and diagnostic conclusions. Menus are displayed in pop-up windows. One of the options on the main menu is an on-line help facility that provides assistance in the use of the menus and graphic displays.

The graphic displays include functional block diagrams of the RE design and physical layout diagrams of the equipment rack and drawers. This is a natural data display approach because the RE operator learns how the RE operates and how to troubleshoot it using the same type of diagrams. On these diagrams, signals that have been monitored or inferred to be good are displayed in green, while those out of tolerance are shown in red. The circuit functions, circuit boards, and rack drawers that are diagnosed as faulty are displayed flashing. Over 50 graphic and tabular displays have been implemented. Figure 6 shows a typical block diagram display and Figure 7 shows a typical rack layout diagram display.

A hypertext-like scheme is used to allow the REDEX user to easily navigate through the set of diagrams and tables. Link "buttons" are defined on each diagram and table. By moving the cursor to one of these link buttons and pressing a function key, a new diagram or table is displayed. These links allow the user to rapidly move up, down, across, and between the hierarchies of block diagrams, layout diagrams, and tabular data. Links between hierarchies are defined in such a way to support the user's thinking process. For example, when the user is viewing a table of RIM point values, he is likely to want to see the location of a given red RIM in the RE. A link button is therefore defined for each RIM point on the table to allow the appropriate subsystem block diagram to be quickly displayed.

5.0 IMPLEMENTATION AND TESTING STATUS

REDEX has reached the advanced prototype stage of implementation. All data interface, user interface, performance, and diagnostic functional requirements specified in the initial requirements document have been satisfied. Over a dozen demonstrations have been given to various review groups, and the feedback obtained has been factored into the evolving prototype. Reviewers have included RE operations personnel, RE design engineers, system engineers, hardware engineers, software engineers, human factors specialists, and artificial intelligence specialists.

REDEX is currently being tested in an offline mode, not directly interfaced with the RE. Testing has been performed using the following three techniques:

(a) Manually constructed test cases - Diagnostic test cases were manually constructed and executed. A test case consists of a set of red RIMs, a set of RE configuration switch settings, and an associated expected diagnosis. Sufficient test cases were designed to exercise each functional diagnostic rule in isolation or in a small set. This testing technique helped identify local coding errors in individual diagnostic rules. It is roughly analogous to structural unit testing of traditional software.

(b) Fault simulator - An RE fault simulator program was developed that simulates the propagation of fault effects through the RE. When the location of a hypothetical fault

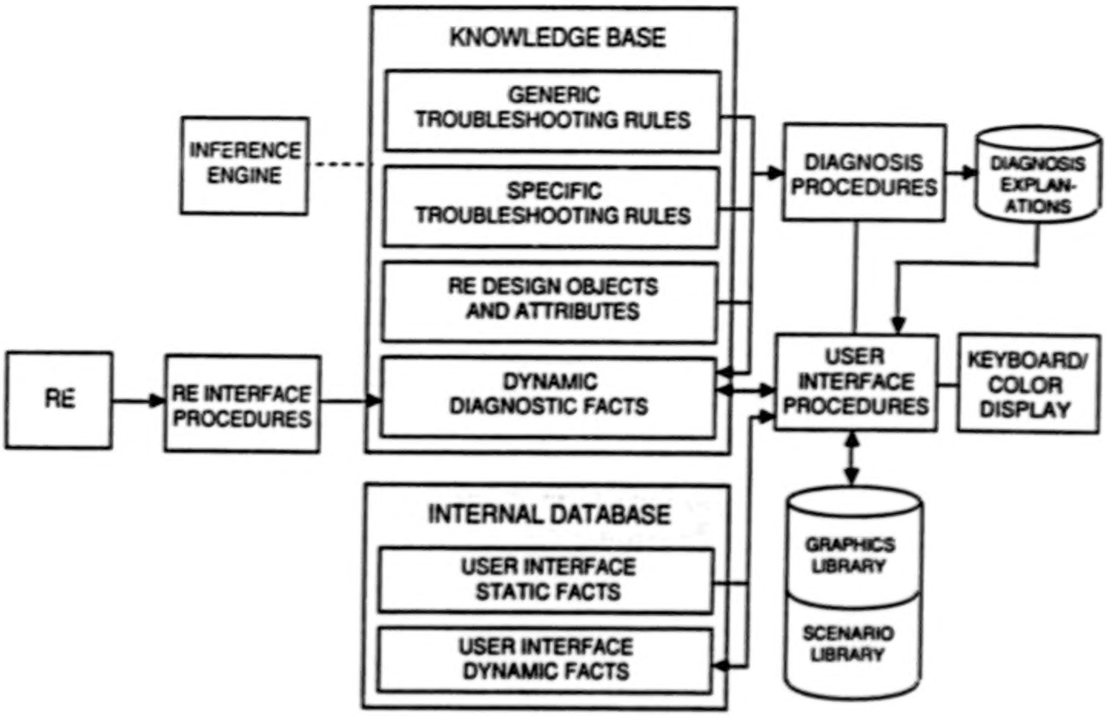

KNOWLEDGE BASE
GENERIC TROUBLESHOOTING RULES
SPECIFIC TROUBLESHOOTING RULES
RE DESIGN OBJECTS AND ATTRIBUTES
DYNAMIC DIAGNOSTIC FACTS
INFERENCE ENGINE
RE
RE INTERFACE PROCEDURES
DIAGNOSIS PROCEDURES
DIAGNOSIS EXPLAN-ATIONS
USER INTERFACE PROCEDURES
KEYBOARD/ COLOR DISPLAY
INTERNAL DATABASE
USER INTERFACE STATIC FACTS
USER INTERFACE DYNAMIC FACTS
GRAPHICS LIBRARY
SCENARIO LIBRARY

Figure 4. REDEX System Block Diagram

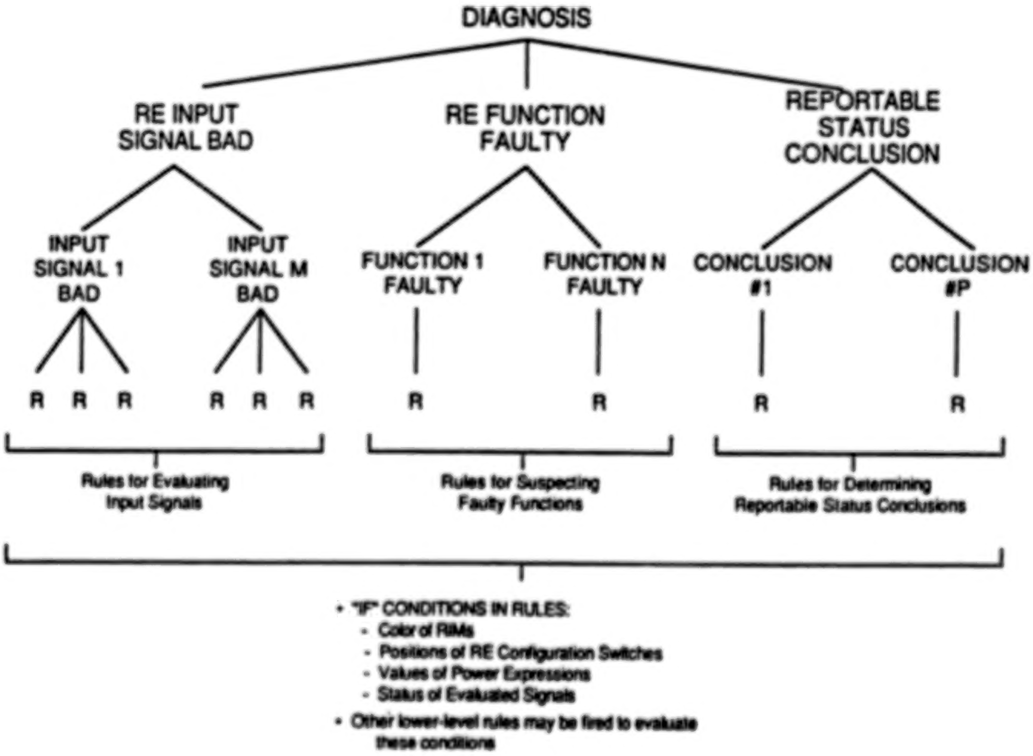

DIAGNOSIS
RE INPUT SIGNAL BAD
RE FUNCTION FAULTY
REPORTABLE STATUS CONCLUSION
INPUT SIGNAL 1 BAD
INPUT SIGNAL M BAD
FUNCTION 1 FAULTY
FUNCTION N FAULTY
CONCLUSION #1
CONCLUSION #P
R R R
R R R
R
R
R
R
Rules for Evaluating Input Signals
Rules for Suspecting Faulty Functions
Rules for Determining Reportable Status Conclusions
• "IF" CONDITIONS IN RULES:
- Color of RIMs
- Positions of RE Configuration Switches
- Values of Power Expressions
- Status of Evaluated Signals
• Other lower-level rules may be fired to evaluate these conditions

Figure 5. RE Diagnosis Fault Tree

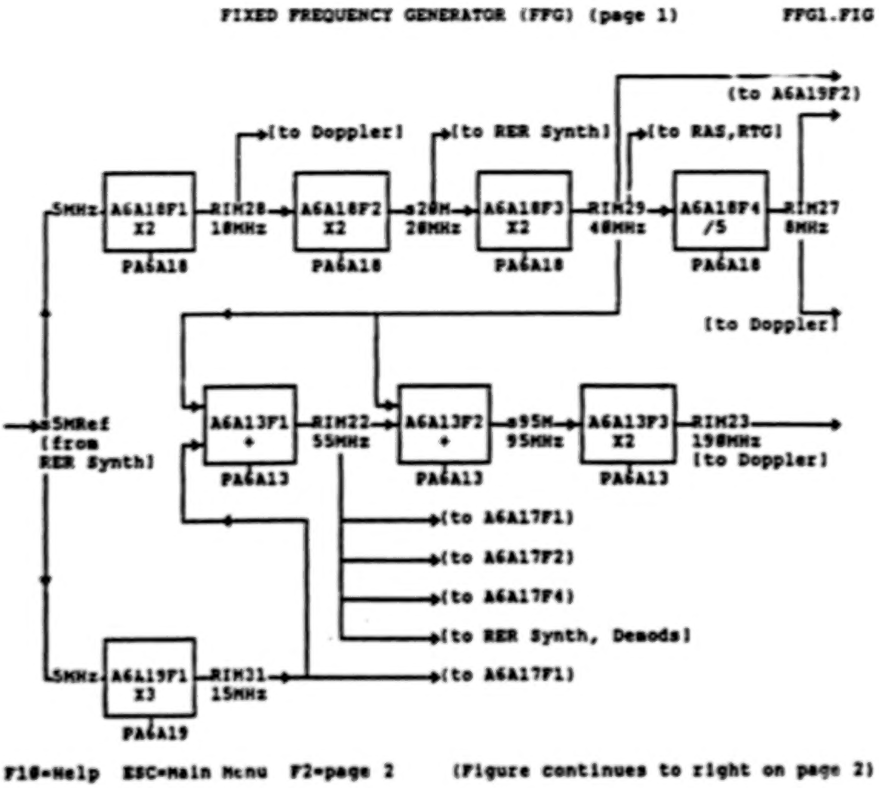

FIXED FREQUENCY GENERATOR (FFG) (page 1)
FFG1.FIG
(to A6A19F2)
[to Doppler]
[to RER Synth]
[to RAS,RTG]
5MHz
A6A18F1
X2
PA6A18
RIM28
10MHz
A6A18F2
X2
PA6A18
s20M
20MHz
A6A18F3
X2
PA6A18
RIM29
40MHz
A6A18F4
/5
PA6A18
RIM27
8MHz
[to Doppler]
s5MRef
[from
RER Synth]
A6A13F1
+
PA6A13
RIM22
55MHz
A6A13F2
+
PA6A13
s95M
95MHz
A6A13F3
X2
PA6A13
RIM23
190MHz
[to Doppler]
(to A6A17F1)
(to A6A17F2)
(to A6A17F4)
[to RER Synth, Demods]
5MHz
A6A19F1
X3
PA6A19
RIM31
15MHz
(to A6A17F1)
F10=Help
ESC=Main Menu
F2=page 2
(Figure continues to right on page 2)

Figure 6. Example REDEX Block Diagram Display

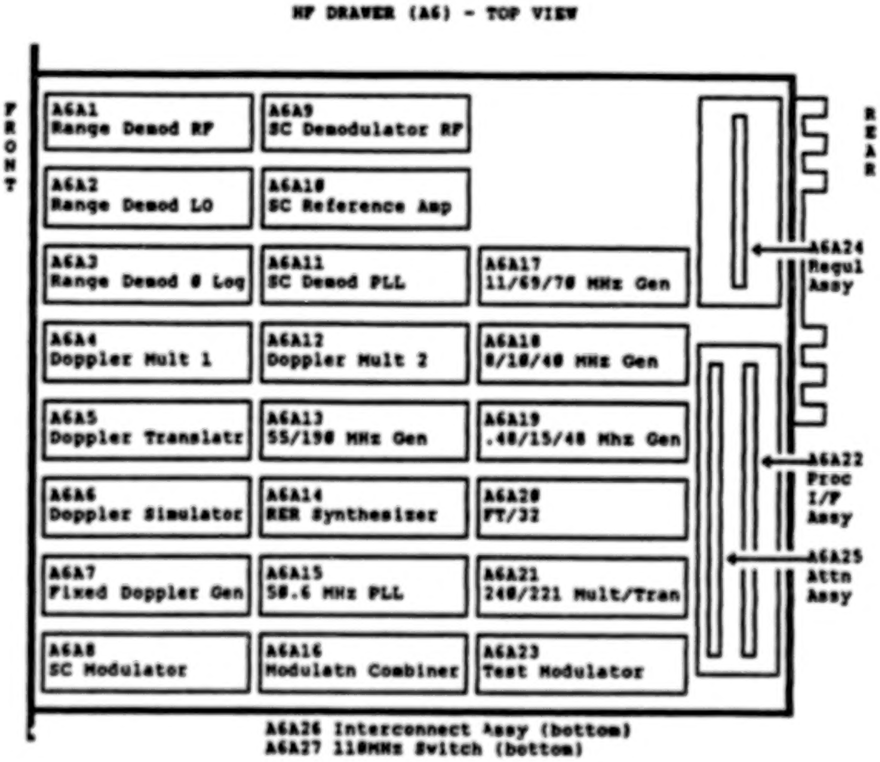

RF DRAWER (A6) - TOP VIEW
FRONT
REAR
A6A1
Range Demod RF
A6A9
SC Demodulator RF
A6A2
Range Demod LO
A6A10
SC Reference Amp
A6A3
Range Demod 0 Log
A6A11
SC Demod PLL
A6A17
11/69/70 MHz Gen
A6A4
Doppler Mult 1
A6A12
Doppler Mult 2
A6A18
8/10/40 MHz Gen
A6A5
Doppler Translatr
A6A13
55/190 MHz Gen
A6A19
.40/15/40 Mhz Gen
A6A6
Doppler Simulator
A6A14
RER Synthesizer
A6A20
FT/32
A6A7
Fixed Doppler Gen
A6A15
50.6 MHz PLL
A6A21
240/221 Mult/Tran
A6A8
SC Modulator
A6A16
Modulatn Combiner
A6A23
Test Modulator
A6A24
Regul
Assy
A6A22
Proc
I/F
Assy
A6A25
Attn
Assy
A6A26 Interconnect Assy (bottom)
A6A27 110MHz Switch (bottom)

Figure 7. Example REDEX Physical Drawer Layout Display

is provided to this program, it generates the set of red RIM points that would result if the effects of the fault propagated fully. This program was used to automatically generate a large number of realistic test cases. An error was uncovered whenever a test case based on a hypothetical fault in function F was diagnosed by REDEX to indicate a fault in some location other than F. (In some cases, the error uncovered is in the fault simulator program, not in REDEX.) This testing technique helped identify errors in diagnostic strategy and in the RE design model in the knowledge base. The simulator should also be a valuable tool for training future RE operators.

(c) RE communications emulator - Because the RE has not yet been available for direct interface testing, an IBM PC was used to emulate the RIM point data transmission function of the RE. This PC was linked to the REDEX workstation using an RS232C communications cable, in the same way that the RE is planned to interface to REDEX. Actual RIM value reports obtained from the RE were entered into the PC, and transmitted over the cable to REDEX. This testing technique helped identify errors in the REDEX communications interface procedures.

REDEX will soon be interfaced with the RE for online testing. After online testing and evaluation is completed, REDEX is planned to be installed at four or more NASA tracking stations.

6.0 CONCLUSIONS

The development of REDEX has led to several observations and conclusions about the design and development methodology for ground system diagnostic expert systems:

(a) Equipment monitor points - The diagnostic strategy in REDEX depends on the availability of the set of status monitoring points within the RE. The internal monitoring system that made REDEX possible was specified as part of the RE design. Such monitoring points would be difficult to retrofit into a system like the RE after it is implemented. When a new equipment system is specified, it is paramount to consider including requirements for an internal monitoring system to provide status data for a diagnostic assistant. Monitor points should be placed at sufficient density to allow high confidence diagnosis to the desired system level (e.g. independent subsystem, or field replaceable circuit board) depending on the sparing and repair philosophy for the equipment. A mechanism for gathering the monitor data and delivering it to the diagnostic assistant, such as via a communications interface, should also be specified.

(b) Knowledge acquisition and requirements definition effort - The amount of time needed for these activities was six months, or about twice that amount originally planned. Despite undesirable schedule delays, it was recognized that careful performance of these activities was essential to the success of the project. Care should be taken when scheduling an expert system development effort to avoid underestimating time for these early activities.

(c) Requirements document - Detailed diagnostic functional requirements for REDEX were specified in a formal requirements document using semantic network diagrams and language-independent rules. The discipline of producing this document was found helpful in crystalizing, evaluating, and verifying requirements. The document was produced and will be periodically updated in parallel with the prototyping effort.

(d) Prototyping - Evolutionary development by iterative prototype enhancement was found to be an effective approach for developing REDEX. Frequent demonstrations of prototype status were important in fueling the evolution, particularly of the user interface.

(e) Testing by fault simulation - The development of the RE fault simulator diverted resources from the development of REDEX, but was assessed to be well worth the cost. The fault simulator was used to generate simulated monitor data for a large number of repeatable fault conditions, and enabled comprehensive testing of REDEX.

(f) Generic troubleshooting rules - The catalog of generic troubleshooting rules compiled for diagnosing RE faults is applicable to other equipment that is similar to the RE. Development of diagnostic assistants for other equipment may be facilitated by using this set of generic rules. The rule set should also facilitate the development of guidelines for effective selection of monitor points required by diagnostic assistants. These guidelines would be helpful in developing future hardware specifications.

(g) User interface - The color coded block diagrams used in REDEX to display monitor point status and diagnostic results have been well-received by system engineers and technicians. This approach has been adopted in another prototype diagnostic expert system for NASA ground equipment.

These observations and lessons learned will influence our continuing and future work.

7.0 FUTURE DIRECTIONS

The work on REDEX may be extended in a number of directions, as described below:

(a) Manual test points - REDEX currently renders a diagnosis based solely on the status of the internal monitor points and the current configuration of the RE. This data is usually sufficient to allow a fault to be diagnosed down to the circuit card level. In some cases, however, the fault can be isolated only down to a set of circuit boards. In these cases, manual observations and tests must be conducted to identify the faulty circuit board. REDEX may be extended to ask the technician for observation data and to recommend manual tests, and use this additional information to sharpen the diagnosis. Manual tests would be recommended considering their cost (in time required to perform), and the potential value of the test results in completing the diagnosis.

(b) Anomalous monitor data - REDEX currently assumes that status monitor data is correct. However, due to the possibility of mis-calibrated or faulty monitoring circuits, there are situations in which monitor data may be incorrect. REDEX may be extended to reason about anomalous monitoring situations, and develop alternate diagnoses based on various interpretations of the monitor data.

(c) Other tracking station equipment - REDEX may be extended to diagnose faults in other components of the RF Systems equipment at the tracking stations, such as the S-Band Exciters and Receivers. This direction would also motivate the extension of REDEX's reasoning with generic troubleshooting rules. REDEX currently uses generic rules to diagnose major portions of the RE, but needs specific rules to handle areas with unusual configurations, or where monitor points are very sparse. A goal would be to develop more powerful techniques for reasoning with generic rules, and thereby eliminate the need for specific rules. If this could be done for a class of equipment, then a diagnostic expert system "shell" could be built containing the full set of generic rules. A diagnostic assistant could be easily built for any equipment item in the class by just populating the knowledge base with a structural and functional description of the equipment item.

(d) Hypertext - The successful application of hypertext techniques in REDEX may be extended to other operator-driven information display functions at the tracking stations. This technique may be an effective approach for providing an integrated operator interface to the various information systems at the tracking stations.

It is felt that the current work on REDEX and these future directions are positive steps in the directions of increasing the levels of automation and performance while decreasing operations costs at the GN tracking stations.

ACKNOWLEDGEMENTS

The authors would like to thank Bikas Das for his assistance in defining the initial prototype, and Andy Rolinski for his review of project documentation. This work was done by Computer Sciences Corporation for Goddard Space Flight Center under contract NAS5-31500, Task Assignment 39-115.

REFERENCES

1. K. Howlin, J. Weissert, and K. Krantz, "MOORE: A Prototype Expert System for Diagnosing Spacecraft Problems," *1988 Goddard Conference on Space Applications of Artificial Intelligence*, May, 1988.

2. B. Cruse and C. Wende, "Expert System Support for HST Operations," *1987 Goddard Conference on Space Applications of Artificial Intelligence and Robotics*, May 1987.

3. J. Bush and S. Weaver, *Gamma Ray Observatory Backup Control Mode Analysis and Utility System (BCAUS) Prototype Requirements, Design, and Implementation*, Computer Sciences Corporation, CSC/TM-88/6069, September 1988.

4. L. G. Hull and P. M. Hughes, "CLEAR: Communications Link Expert Asssistance Resource," *1987 Goddard Conference on Space Applications of Artificial Intelligence and Robotics*, May 1987.

5. Kennedy Space Center, "Liquid-Oxygen Expert System," *NASA Tech Briefs Technical Support Package*, KSC-11332.

6. S. A. Mouneimne, "Mission Telemetry System Monitor: A Real-Time Knowledge-Based System," *1988 Goddard Conference on Space Applications of Artificial Intelligence*, May, 1988.

7. R. Dominy, "A Local Area Computer Network Expert System Framework," *1987 Goddard Conference on Space Applications of Artificial Intelligence and Robotics*, May 1987.

8. D. Lowe, et. al., "FIESTA: An Operational Decision Aid for Space Network Fault Isolation," *1987 Goddard Conference on Space Applications of Artificial Intelligence and Robotics*, May 1987.

9. W. Wilkinson, et. al., "Achieving Real-Time Performance in FIESTA," *1988 Goddard Conference on Space Applications of Artificial Intelligence*, May, 1988.

10. D. J. Zillig, "S-Band Ranging System for NASA Mission Support," *Proceedings of the 1987 International Telemetering Conference*, San Diego, CA, October 26-29, 1987.

11. J. Baumert, A. Critchfield, and K. Leavitt, "The Need for a Comprehensive Expert System Development Methodology," *1988 Goddard Conference on Space Applications of Artificial Intelligence*, May, 1988.

12. E. Luczak, *Diagnostic Expert System for the GN Ranging Equipment - Requirements Document*, Computer Sciences Corporation, July 1988.

13. E. Luczak, *Diagnostic Expert System for the GN Ranging Equipment - REDEX Design*, Computer Sciences Corporation, October 1988.

14. L. Gilstrap and J. Retter, *Expert System Development Methodology Users Guide*, Computer Sciences Corporation, CSC/TM-88/6146, September 1988.

15. L. Gilstrap and J. Retter, *Expert System Development Methodology Standard*, Computer Sciences Corporation, CSC/TM-88/6147, September 1988.

16. E. Luczak, *Diagnostic Expert System for the GN Ranging Equipment - Prototype Tool Evaluation Report*, Computer Sciences Corporation, September 1988.

BLANK PAGE

DIAGNOSTIC TOLERANCE FOR MISSING SENSOR DATA

Ethan A. Scarl

Boeing Computer Services
P.O.Box 24346 Seattle, WA 98124-0346

Abstract

For practical automated diagnostic systems to continue functioning after failure, they must not only be able to diagnose sensor failures but also be able to tolerate the absence of data from the faulty sensors. We show that conventional (associational) diagnostic methods will have combinatoric problems when trying to isolate faulty sensors, even if they adequately diagnose other components. Moreover, attempts to extend the operation of diagnostic capability past sensor failure will necessarily compound those difficulties. Model-based reasoning offers a structured alternative that has no special problems diagnosing faulty sensors and can operate gracefully when sensor data is missing.

Introduction

The success of space missions will depend critically upon the robustness of space systems' abilities to monitor, diagnose, and compensate for faults. Since sensors are at least as likely to fail as other components, an unforeseen loss of sensor information should degrade onboard diagnostic capabilities as little as possible. Ideally, diagnostic performance should degrade only to the extent that the information needed to isolate a fault is unavailable, and never because they are incapable of using the information which is still available in the altered environment.

Autonomous systems may be confronted by missing data for many reasons. For example, sensor polling machinery can fail, certain conditions may render a given sensor untrustworthy or invalid (e.g., the wrong fluid may be passing over a thermometer, or a radar may be tracking the wrong object), or a filter may reject the reading as spurious. The most common case, however, is the loss of data due to unpredicted sensor failure.

Knowing how to operate with missing data is distinct from the problem of recognizing the loss of a working sensor (diagnosis). It must be *known* that the data is missing. Both conventional methods and model-based reasoning may reach wrong conclusions if they try to make inferences from obsolete data instead of knowing that the data is not available. For some conditions, like a glitch in the sensor polling machinery, it is the responsibility of the supporting and interface systems to notify the diagnoser. On the other hand, the diagnoser will be the source of knowledge of a sensor failure, and may well have deduced the presence of an inappropriate sensor mode or environment.

Model-based reasoning permits a diagnostic algorithm that reasons explictly about the behavior of physical or computational system components and the connections between them. It will be seen that this algorithm approaches the ideal utilization of available information and does not suffer from combinatoric problems which afflict conventional methods trying to accommodate missing data from sensor failure.

The method suffers only from the requirement that system knowledge be properly represented and manipulated. Contrary to some common misconceptions, it is not necessary to have a model that is complete to any particular level of detail.

The next section describes the associational diagnostic approach most commonly used today, and which includes much of the work in rule-based expert systems as well as more traditional implementations of diagnostic logic with procedural languages. (Readers familiar with the short-comings of conventional methods may wish to skip over this section.) The last section summarizes the model-based approach to diagnosis and then contrasts it with associational methods in its ability to diagnose sensor failure and to tolerate missing data.

Conventional Methods

Traditional approaches have tended to see diagnosis as the analysis of sensor data to determine the most likely faults. Automated diagnostics are usually implemented by rules or procedures which associate faults with patterns of sensed observation; hence the term *associational.*

Some of these inference methods chain forward directly from sensor data to faulty component likelihoods using an arbitrary distillation of expertise (rule- or table-driven associational methods), while others postulate faults, (manually) determine the resulting patterns of sensor readings, and finally reverse-index these faults by symptom (Failure Modes and Effects Analysis, or FMEA methods). In either case, the operating logic is to take sensor inputs and match them against patterns that indicate specific failures or classes of failure.

A matched pattern associates the present system condition with a predetermined failure. A successful match means that a fault has been located (or at least deemed likely), while an unsuccessful match means only that that particular fault mode is unlikely. Only after *all* fault modes have *failed* to match can the system be assumed to be operating properly. This is such an ungainly method for monitoring a healthy system that a number of applications have used model-based simulation (below) for monitoring, only to fall back on associational methods for fault location [e.g., Hofmann 88; Zwinglestein 85].

The implementation of an associational diagnostic inference may use production rules, a decision table, a programmed procedure, a decision tree, or an FMEA tree, but we will think of the knowledge it contains as equivalent to a collection of rules. A typical such rule might say that:

"if
  sensor A = 0, sensor B ε (100, 150), sensor C > 10, and sensor D = ON,
 then
  there an 80% chance that component E is broken and a 20% chance that component F is broken."

One might add:

"else
  E and F are probably working, and there may even be nothing wrong with anything."

Broken Sensors

The first major problem is that this rule depends upon sensors A, B, C, and D all working properly. If any of them is faulty, the rule will deduce a wrong conclusion.

In principal, sensor failures would be inferred by similar rules, specially written for that purpose. These rules for determining sensor health are special in that they must be run *before* the above system health rule to prevent it from running with bad data; [Fox 83] refers to these as *meta-rules*.

As a simpler example, consider determining the health of four totally redundant sensors (call them P, Q, R, and S) which all measure the same parameter X. With perfect redundancy, it is possible to use simple "voting." A voting meta-rule might be:

"if P = Q, and Q = R, and P ≠ S, then S is faulty,"

Unfortunately, three more such rules are needed to determine the guilt of P, Q, and R. Note that four rules, one per sensor, are sufficient only because it is being assumed that no other sensors are relevant to X. Additional sensors would not only add more rules but would add complexity to the rules just given.

It is usually too burdensome to continually rerun all of these fault isolation rules just to be sure that everything is working. A speedup (if not a simplification) to determine that *nothing* is broken would be to use a special (hyper-meta- ?) rule for monitoring:

"if
  P = Q and Q = R and R = S,
then
  none of P, Q, R, or S is broken, so skip their fault isolation meta-rules and go directly to the rules that use P, Q, R, and S to diagnose other components."

Totally redundant sensors are not often available, and information from sensors that yield different but related quantities must be fused by less obvious meta-rules to infer the poor health of each in turn. For each sensor, another meta-rule must be written for each of the ways that its failure can be identified. This is rarely attempted for complex systems, and serious attempts have ended in failure due to the sheer number of cases that must be analysed, programmed, and validated [Delaune 85; Jamieson 86].

In summary, conventional associational methods have serious methodological and combinatoric difficulties in coping with sensor failure. This problem is often misleading to software developers, since they tend to feel that if each condition is coded methodically enough, or they just debug the most recent unexpected rule interactions, they will succeed. Unfortunately, the problem is inherent in the associational approach. It is also worth noting that relatively minor subsequent changes made to the system may require substantial changes to the associational diagnostic code, with considerable effort expended in revalidation.

Missing Data

Missing sensor data corresponds to variables in the hypotheses of sensor-fault meta-rules being unbound. What happens to the inference that

"if P = Q, and Q = R, and P ≠ S, then S is faulty,"

after sensor P stops working? One possibility is that no rule using P is permitted to fire, so that an additional meta-rule like

"if Q = R, and Q ≠ S, then S is faulty,"

is needed for every possible combination of sensed data being present or absent, along with corresponding methods for propagating uncertainty, for every meta-rule. (Remember that the number of meta-rules is already likely to be a serious problem.) The only other alternative is to let the meta-rules fire in spite of missing data; they then must bear the responsibility of internally testing for all the possible combinations of missing data, and responding appropriately. In either case, special code must be written in advance for each such combination.

One may attribute associational reasoning's inadequate handling of breaking or broken sensors to its unprincipled mixing of structural and behavioral knowledge with environmental knowledge (sensor readings). This can force a great many repetitive representations of the same knowledge.

For example, consider again the four identical sensors P, Q, R, and S. Since the isolation of multiple faults involves combinatoric problems by any method (and since these combinatorics are more obviously expressed in the rules of associational systems than in the algorithms of model-based reasoning), let us consider the number of rules required for isolating a single fault and then for continued operation after multiple faults have occurred. This is not altogether unreasonable since multiple faults that did *not* happen simultaneously, and therefore can be diagnosed as single faults, will have a cumulative effect on continued operation. Our four sensors then require one monitoring rule, four diagnostic rules, 15 rules to cover all possible combinations of multiple breakage among sensors P, Q, R, and S in the monitoring rule, and 7 additional rules for *each* diagnostic rule (e.g., how to prove that S may be faulty if one or more of P, Q, and R are already known to be broken). This means that a total of 48 rules need to be written to support the continued diagnosis of this configuration after failures.

Of course, an actual associational architecture would try to minimize this by combining the repetitive representations in some way. We predict that such compaction will be achieved either through *ad hoc* mechanisms like deleting all conjunctive or disjunctive terms that mention a defunct sensor, or through references to something equivalent to a model-based representation.

The combinatorics of missing sensor data compound the combinatorics of sensor diagnosis. The traditional methods for developing and maintaining monitoring/diagnostic software are costly, time consuming, and incapable of handling situations that were unforeseen when the system was written. We conclude that associational methods are unsuitable for the complete and robust diagnosis of complex systems after failure.

Model-Based Generate and Test

An alternative approach has been (with differing nuances) variously referred to as "causal reasoning," "deep reasoning," "reasoning from first principles," and "reasoning from knowledge of structure and function." More recently, it has been simply called "model-based reasoning," because the "depth," "first principles," or "knowledge of structure and function" were invariably embedded in a model which (at some level of detail or abstraction) was isomorphic to the system, both in structure and behavior (function). This approach is described elsewhere [Davis 85, 88; Genesereth 85; Scarl 87, 88] in a variety of implementations, but will be summarized here.

The view embodied here is that all system components are initially under suspicion as possible causes of a perceived malfunction, and the job of *diagnosis* is to eliminate inconsistent suspects by showing that the assumption that they are responsible is contradicted by sensory observation. In model-based reasoning, the system is represented by some network of significant components or computational quantities whose outputs are determined by transfer functions upon their inputs. Suppose, for example, that components A and B have outputs connected to the inputs of component C, and that the output of C is directly measured by sensor S. When the states of A and B have been determined by setting their inputs, then their outputs determine the inputs to C, whose output in turn determines S.

The original conception was that these components corresponded one-to-one to physical system components, and that the structure of the model corresponded one-to-one to the connections between the physical components. This is often still true, but it has become clear that some systems need to have local properties (e.g., the voltage across a resistor) defined in terms of more global properties that are derived from the compaction or simultaneous analysis of different parts of the system (e.g., the current through a series circuit). This leads to the introduction of objects which represent global abstractions (e.g., total path resistance) rather than physical components.

An inversion facility is also required: given a component's output (by measurement, inference, or assumption), what can we say about its inputs? Usually (especially when a single point of failure is being sought), the inputs can be considered one at a time, with the others assumed to be as computed from the system commands. This uses a very broad sense of "inversion." If the component's transfer function is a "trapdoor function" that cannot be practically inverted, so that the output tells us nothing about the input, then the "inverse" is its whole domain. The trick is to be able to represent *whatever* information is thus provided.

The model is a behavioral simulation which can be used to monitor the system's operation. Choosing a set of commands (system inputs) for the model causes predictions to be computed for the system's sensors. Although the health of the system is again determined by matching sensor readings against these prescribed values, there are important differences from the associational approach:

- A match means a healthy system, instead of a fault

- The set of sensors to be matched can be computed dynamically and interactively

- The matching is done against values computed dynamically from system commands, rather than statically predetermined. (Note that associational rules could also compute the comparison value dynamically from commands, but, if so, they would be performing model-based reasoning rather than associational reasoning.)

Any discrepancy between predicted and measured values indicates either a failure in the physical system or an inaccuracy in the model. We will assume in this discussion that the model accurately describes the system's proper behavior.

Faults are located by generating hypotheses and testing those hypotheses against all available sensor data. Hypotheses usually are generated directly from sensor readings by using the inversion of the functional relationships in the model. A hypothesis typically concerns some particular object C, and has the form:

"The inputs of component C have the values expected for them, but C's output O is broken so that it has the unexpected value (or range of values, if analog) X."

For example, if the observed value of a discrepant sensor S is passed through inverted functionalities to component C, then the resulting hypothesis is a single fault in component C with the specific wrong value X at its output. C is then a *suspect* for the failure consistent with S. If C is the only suspect, then C has been determined to be the single point of failure, as determined by this model.

If a hypothesis is generated by simultaneously using all sensors, then it needs no further validation; since all available observational information has been used to manufacture it, there is none left that could contradict it. Usually, however, the hypothesis is generated from the reading of a single *discrepant* sensor that disagrees with the model's prediction, and all the other related sensors are used to verify the hypothesis. Whatever mechanism is used to generate a hypothesis, the hypothesis is testable in the model by inserting its hypothesized value (or range) for O in place of O's expected value.

Broken Sensors

Model-based reasoning has no need of any additional knowledge (such as meta-rules) to tell whether sensors are broken, but is able to simply treat a sensor like any other component [Scarl 88]. A sensor with a discrepant reading is *always* included as a suspect. A hypothesis will be generated to say that the sensor is broken so as to read what it actually did rather than what the model predicted for it. That hypothesis is tested for consistency with other sensor readings, just as any other component would be, and rejected or retained accordingly.

Thus, no special rules are needed just because a component is a sensor. Nor need sensors be cleared of suspicion before testing other objects. Instead, the hypotheses generated for *all* components are tested (simultaneously or in parallel) against current sensor readings, which, after all, constitute *all* the useful knowledge available, and the *only* source of available knowledge (in addition to the model's knowledge of system structure and function). The inherent parallelism of the model-based approach is a distinct advantage: a faulty sensor hypothesis can be tested in parallel with hypotheses about other objects.

The complexity of diagnosing sensors is therefore no greater than diagnosing other types of component. In fact, certain assumptions can simplify the generate-and-test algorithm for sensors to the point of being trivial [Scarl 88]. This happens when the structure is such that sensors cannot be responsible for each other's discrepancies. If a maximum of N simultaneous faults is assumed, then more than N simultaneously discrepant sensors will validate each other. Thus, each sensor S that gives a discrepant reading is the *only* sensor in its associated suspect list (in the language of [de Kleer 87], its minimal conflict set). The existence of N other discrepant sensors will clear S of participation in an N-tuple failure.

This shortcut does not apply if sensed values are used for control (feedback), but it *does* apply to sensors whose control is independent of their actual readings. For example, an ammeter will control the current through its circuit by virtue of being in series, but it can be separated into a resistor and a virtual sensor. It is then its resistor and not its sensor that controls other sensors in the circuit, and any other discrepant sensor will clear the ammeter of being a single point of failure.

In summary, the model-based reasoning approach to finding faults in sensors is no more difficult than finding faults in other objects, and does not become more complex as more sensors are added to the system.

Missing Data

When data is lost, for any of the reasons mentioned in the Introduction, model-based reasoning takes account of that loss in rather obvious ways. First, we do not monitor the missing data, and so it cannot trigger diagnosis. Second, we simply exclude missing data from the pool of data against which failure hypotheses are tested. This is similar to deleting terms referring to faulty sensors from associational rules, but simpler and free of *ad hoc* character.

Any number and combination of sensor failures can therefore be handled in a straightforward and uniform way, with no need for special coding.

Returning to the four-sensor example from the discussion of associational diagnostics, if some fault is hypothesized which affects the predicted value of parameter X, then only those sensors known to be operational are used to confirm that hypothesis.

Recall that while only four diagnostic rules were required (and that only because no other system sensors were considered) by an associative diagnoser, 48 rules were required to continue diagnosis after sensor failures. Model-based reasoning, on the other hand, will perform all inferences required for continued operation using only four functional descriptions, one for each sensor, plus the information that they are all connected to X. Each functional description simply declares that its sensor's predicted reading is given by the value of its input parameter. In the KATE representation [Scarl 88], this functional description would simply be the name of (the output of) some object which is supposed to set parameter X. Furthermore, if all four sensors are identical, the four functional descriptions need not be written manually but may be inherited from a generic sensor type description.

The model-based method of handling missing data is conservative, in that fault hypotheses may be retained which could have been ruled out were the data available, but there is no possibility of wrongly abandoning a valid hypothesis.

In conclusion, a brief analysis has shown that traditional associative approaches to diagnosis cannot be extended effectively to operate after sensor failure, not only because structure and function is not explictly represented, but also due to the sheer numbers of redundant representations of the knowledge that is present. The model-based generate-and-test algorithm elegantly avoids these difficulties and promises much more robust diagnostic capabilities.

References

[Davis 85] R. Davis, "Diagnostic Reasoning Based on Structure and Behavior," in D. G. Bobrow (ed.), Qualitative Reasoning about Physical Systems, MIT Press, Cambridge, MA, 1985, pp. 347-410.

[Davis 88] R. Davis and W. Hamscher, "Model-based Reasoning: Troubleshooting," in H. E. Shrobe (ed.), Exploring Artificial Intelligence: Survey Talks from the National Conferences on Artificial Intelligence, Morgan Kaufmann, San Mateo, CA, 1988, pp. 297-346.

[de Kleer 87] J. de Kleer and B. C. Williams, "Diagnosing Multiple Faults," *Artificial Intelligence*, **32**, No. 1, April, 1987, pp. 97-130.

[Delaune 85] C. I. Delaune, E. A. Scarl, and J. R. Jamieson, "A Monitor and Diagnosis Program for the Shuttle Liquid Oxygen Loading Operation," *Proceedings of the First Annual Workshop on Robotics and Expert Systems*, Houston, TX, June, 1985.

[Fox 83] M. S. Fox, S. Lowenfeld, and P. Kleinosky, "Techniques for Sensor-Based Diagnosis," *Proceedings of the Eighth International Joint Conference on Artificial Intelligence (IJCAI-83)*, August, 1983, pp. 158-163.

[Genesereth 85] M. Genesereth, "The Use of Design Descriptions in Automated Diagnosis," in D. G. Bobrow (ed.), Qualitative Reasoning about Physical Systems, MIT Press, Cambridge, MA, 1985, pp. 411-436.

[Hofmann 88] A. G. Hofmann, J. G. Allard, L. B. Hawkinson, and M. Levin, "Object Oriented Simulation Models and the Uses in Real Time Expert Systems," *Proceedings of the Third Artificial Intelligence and Simulation Workshop at AAAI-88*, St. Paul, MN, August, 1988.

[Jamieson 86] J. R. Jamieson, "A Mandate for Autonomous Control and Monitor Systems (The Failure of Hard Automation)," 1986, *available from the author.*

[Scarl 87] E. A. Scarl, J. R. Jamieson, and C. I. Delaune, "Diagnosis and Sensor Validation through Knowledge of Structure and Function," *IEEE-Transactions on Systems, Man, and Cybernetics*, **SMC-17**, No. 3, May/June, 1987, pp. 360-368.

[Scarl 88] E. A. Scarl, J. R. Jamieson, and E. New, "Deriving Fault Location and Control from a Functional Model," *Proceedings of the Third IEEE Symposium on Intelligent Control*, Arlington, VA, August, 1988.

[Zwinglestein 85] G. Zwinglestein, J. L. Tyran, P. Bajard, "Failure Detection Based on Multilevel Dynamic Models Applied to a U-Tube Steam Generator for a PWR," *Proceedings of the Symposium on New Technology in Nuclear Power Plant*

Image Processing and Machine Vision

BLANK PAGE

Robot Acting on Moving Bodies (RAMBO): Preliminary Results

Larry S. Davis, Daniel DeMenthon, Thor Bestul, Sotirios Ziavras, H.V.Srinivasan,
Madhu Siddalingaiah, and David Harwood

Computer Vision Laboratory
Center for Automation Research
University of Maryland, College Park, MD 20742

Abstract

We are developing a robot system (RAMBO) equipped with a camera, which, given a sequence of simple tasks, can perform these tasks on a moving object. RAMBO is given a complete geometric model of the object. A low level vision module extracts and groups characteristic features in images of the object. The positions of the object are determined in a sequence of images, and a motion estimate of the object is obtained. This motion estimate is used to plan trajectories of the robot tool to relative locations nearby the object sufficient for achieving the tasks.

More specifically, low level vision uses parallel algorithms for image enhancement by symmetric nearest neighbor filtering, edge detection by local gradient operators, and corner extraction by "sector filtering". The object pose estimation is a Hough transform method accumulating position hypotheses obtained by matching triples of image features (corners) to triples of model features. To maximize computing speed, the estimate of the position in space of a triple of features is obtained by decomposing its perspective view into a product of rotations and a scaled orthographic projection. This allows us to make use of 2D lookup tables at each stage of the decomposition. The position hypotheses for each possible match of model feature triples and image feature triples are calculated in parallel. Trajectory planning combines heuristic and dynamic programming techniques. Then trajectories are created using parametric cubic splines between initial and goal trajectories. All the parallel algorithms run on a Connection Machine CM-2 with 16K processors.

1 Introduction

The problem of robotic visual navigation has received considerable attention in recent years, but research has mostly concentrated on operations in static environments [1–11]. The area of robotics in the presence of moving bodies has seen little activity so far [12–16]. We are developing a control system which should allow a robot with a camera to accomplish a sequence of actions on a moving object.

We have set up an experimental facility which has the necessary components for testing various vision-based control algorithms for intercepting moving objects. This facility is described in the following section.

2 Experimental set-up

A large *American Cimflex* robot arm, RAMBO, is equipped with a CCD camera and a laser pointer (Figure 1, top left). Images from the camera are digitized and sent to the Connection Machine for processing. A smaller robot arm (*Mitsubishi RM-501*) translates and rotates an object (called the *target* in this paper) through space. Several light-sensitive diodes with focusing optics are mounted on the surface of the object. RAMBO's goal is to hit a sequence of diodes on the moving object with its laser beam for given durations, possibly subject to overall time constraints. Electronics inside the object signal success by turning on an indicator light. Simultaneously, we are developing a full computer simulation in which the camera inputs are replaced by synthetic images (Figure 1, top right).

3 Summary of Operations

The vision-based control loop for RAMBO is shown in Figure 1. We briefly describe the functions of the different modules of this system from data collection to robot motion control, and refer to the sections of this paper which give more details.

1. The digitizer of the video camera mounted on the robot arm can grab video frames when new visual information is needed. A database contains a list of positions of feature points on the target, in the local coordinate system of the target.
2. A low-level vision module extracts locations of feature points from the digitized image (Section 4).
3. An intermediate vision module finds the location/orientation of the target in the camera coordinate system (Section 5, Appendix A and Appendix B).
4. Since the past camera trajectory is known, the position of the camera in the robot base coordinate system when the frame was grabbed is known. The location/orientation of the target is transformed to the robot base coordinate system (Section 6).
5. This most recent target pose at a specific time is added to the list of target poses at previous times. In the Target Motion Predictor, a target trajectory in location/orientation space is fitted to these past target poses and extrapolated to the future to form a predicted target trajectory. We also obtain the predicted trajectories of *goal points* around the target. A goal point is a location –which is fixed in the frame of reference of the target, thus moving in the frame of the robot base– that one of the joints of the robot has to follow for the accomplishment of one of the subtasks of the total action (Section 6).
6. From the predicted goal point trajectories, the Robot Motion Planner calculates the robot motions necessary for following the goal points, and the resulting camera trajectories (Sections 7, 8, 9). If the subtasks are not ordered, the Motion Planner finds an optimal order (Section 10). The camera trajectories are used for transforming subsequent target pose estimates from a camera coordinate system to an absolute coordinate system (Section 6).

4 Low-level Vision

The model-based pose estimation described in the next section requires that feature points be extracted from each of the images of the target. This is the task of the low-level vision module. Feature points in an image could be images of small holes in the target structure, corners of letters, vertices, etc, ...Conversely, the geometric description of the target should contain the 3D locations of the feature points which are easily detected in images.

In our experiments, the target is a polyhedra, and the feature points that we use are the vertices of the polyhedra, so that our image analysis is partly specific to this type of feature detection. Our image processing algorithms involve a sequence of basic local operations [17, 18] implemented on the Connection Machine –enhancement [19], edge detection, edge thinning and vertex detection– followed by some simple processing to determine which pairs of vertices are connected by edges in the image. This last operation proceeds as follows:

Given k vertices, we use the processing cells in the upper-triangle of a $k \times k$ array of cells in the Connection Machine, each assigned to a possible edge between vertices.

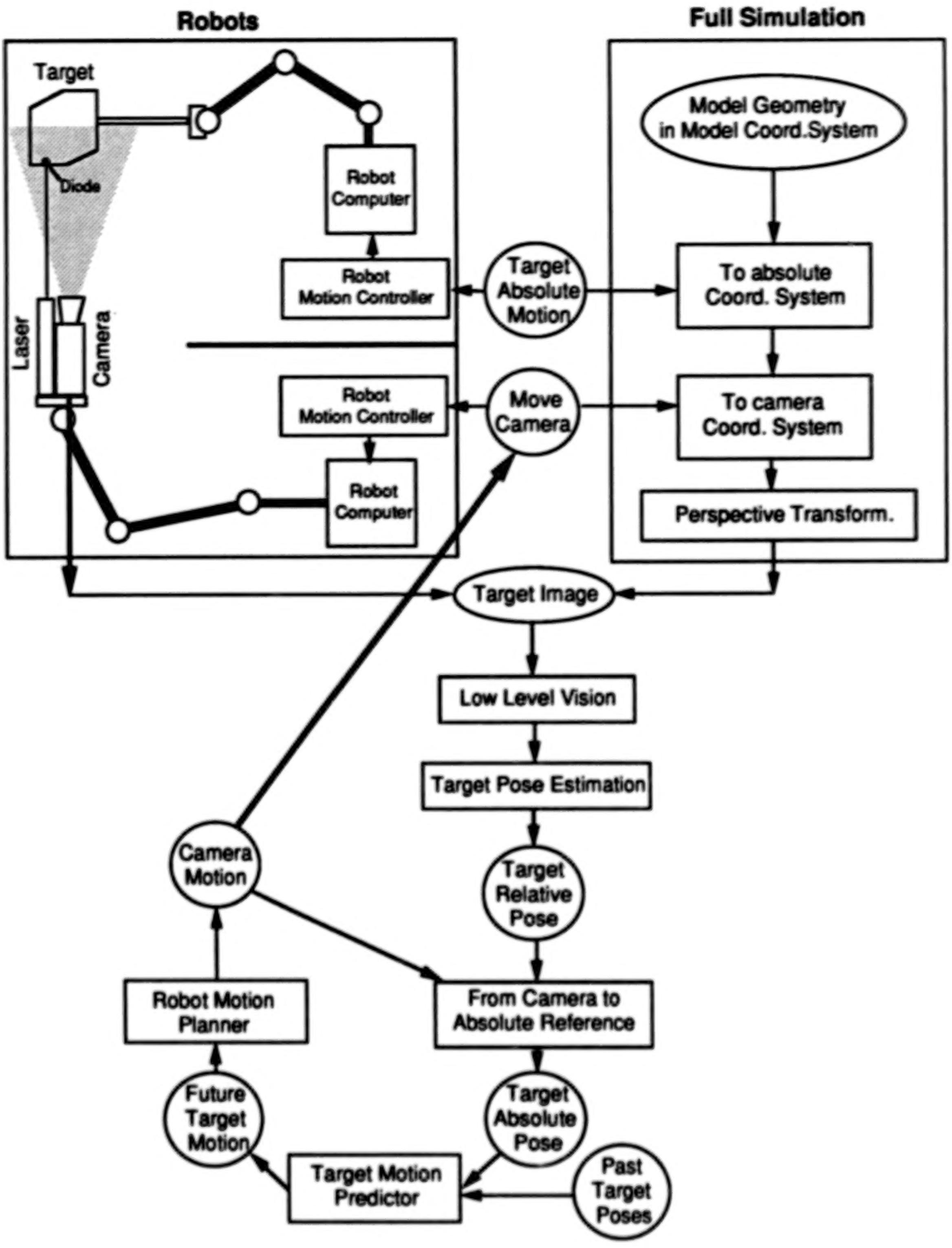

Robots
Full Simulation
Target
Diode
Robot Computer
Robot Motion Controller
Target Absolute Motion
Laser
Camera
Robot Motion Controller
Move Camera
Robot Computer
Model Geometry in Model Coord.System
To absolute Coord. System
To camera Coord. System
Perspective Transform.
Target Image
Low Level Vision
Target Pose Estimation
Camera Motion
Target Relative Pose
Robot Motion Planner
From Camera to Absolute Reference
Future Target Motion
Target Absolute Pose
Target Motion Predictor
Past Target Poses

Figure 1. Vision-based control loop for a robot acting on a moving body

1. Enable a grid of $k \times k$ cells. Disable cells in the lower triangle of the array.

2. Copy the addresses of corner points and the incident angles of their edges to the cells in the diagonal of the grid.

3. By horizontal grid-scan and vertical grid-scan, spread the incident angles and the addresses of the vertices from the diagonal cells along rows and columns to all the cells in the array.

4. Each non-diagonal active cells (i, j) now has all the information about the i-th and j-th vertices; these cells can determine whether they have a pair of collinear incident edges. If they do, then that vertex pair is marked as being connected (Note that we could also count the number of edge pixels along the line joining the vertices, but this would be much more costly on the Connection Machine and not worthwhile for our purposes).

From this algorithm we obtain a list of all the vertices with for each vertex a sublist of the vertices connected to it. We then produce a list of all the image triples consisting of one vertex with two vertices connected to it. This list of image triples is input to the intermediate-level vision module described in the next section. The other input is a similar list for the triples of world vertices of the target, from the geometric database describing the target.

5 Intermediate-level Vision: Pose Estimation of the Target

The pose estimation algorithm combines three ideas:

1. Pose estimation by matching triples of image features to triples of target features [20].

2. Standard camera rotations [21].

3. Paraperspective approximation to perspective projection [22, 23].

This combination allows the extensive use of 2D look-up tables to replace the costly numerical computations used by similar previous methods [20, 24–26]. The algorithm is implemented on the Connection Machine.

The feature points detected in an image are grouped into triples (the *image triangles*). Each image triangle can be described by one of its vertices (the *reference vertex*), the length of the two adjacent sides, and the angle between them (the *reference angle*). These adjacent sides do not necessarily have to correspond to actual edges in the image, and all distinct triples of points could be considered, with each triple of points producing three such image triangles. However, if the feature points are vertices, it is useful to only consider image triangles in which the adjacent edges of the reference vertex are actual edges, and to only match these image triangles to world triangles with similar characteristics. This increases the proportion of good matches over the total number of possible matches.

The main algorithm steps are as follows:

1. Each image triangle is transformed by a *standard rotation*. The standard rotation corresponds to the camera rotation around the center of projection which brings the reference vertex of the triangle to the image center. For each reference vertex in the image, rotation parameters are read in a 2D lookup table.

2. Image triangles are then rotated in the image plane around the reference vertex (located at the image center) to bring one edge into coincidence with the image x-axis.

3. Once in this position, an image triangle can be described by three parameters only, the reference angle, the edge ratio (ratio of the lengths of the two edges adjacent to the reference vertex), and a size factor.
4. For each image triangle/target triangle pair, a 2D lookup table can be used to determine the orientation of the target triangle in space. There is one 2D lookup table per target triangle, which gives two possible orientations of this target triangle when the reference angle and edge ratio of its image are entered. Details on the calculations of these tables, based on a paraperspective approximation, are given in Appendix A.
5. Comparing the size of the image triangle to the size of the target triangle of known orientation, we can then find the distance of the target triangle from the camera lens center.
6. The preliminary transformations of the image triangle can be reversed to obtain the actual 3D pose of the target triangle, the corresponding 3D position of the target center, and the image of the target center.
7. The target center projections are clustered to identify the pose of the whole target.

When RAMBO analyzes its first image, it does not have any *a priori* knowledge of which feature triangles are visible. In this case, the system uses all the possible combinations of target triangles and image triangles. However, clustering gives better results if most improper matches are removed, and it is possible to do so after a few consistent pose estimates of the target have been obtained. The system can also avoid considering matches for target triangles which are at a nearly grazing angle with the lines of sight, since for these triangles image analysis is likely to perform poorly and paraperspective does not approximate true perspective well.

Details on the implementation of this pose calculation method on the Connection Machine are given in Appendix B. For a target producing less than 16K image triangle/target triangle combinations, each pose calculation takes around one second on a CM-2 with 16K processors but without floating point processors.

6 Motion Prediction

The computation of a target position from an image gives the translation vector and rotation matrix of the target coordinate system in the camera coordinate system. However, the camera itself is set in motion by the robot arm. The trajectory of the camera in an absolute coordinate system is known, and it is straightforward to get the position of the camera coordinate system at the time the image was taken and to find the target position at this time in an absolute coordinate system.

From a sequence of target positions, the robot must be able to predict future positions of the target in order to construct plans of actions. These target positions are points in six-dimensional space (three translation parameters and three rotation angles), each with a time label. We can fit a parametric function of time, such as a polynomial, to each of these sequences of coordinates. The target trajectory is then described parametrically by six functions of time. Calculating these functions for a future value t of the time parameter will give a predicted target position at this future time.

7 Task and Trajectory Planning

In order to perform task and trajectory planning, RAMBO currently makes the simplifying assumption that a complex goal can be decomposed into a sequence of simple subgoals, and that each subgoal can be performed with one joint of the robot in a fixed position with respect to the target. This joint has to "tag along" with the target, thus we call this joint the *tagging joint* of the robot. The fixed position with respect to the target that the tagging joint must follow to complete a subgoal will be called a *goal point*. All goal points required for each complex action on a target can be predefined in a data base of actions specific to each target. Each goal point is defined by six coordinates, three for the location and three for the orientation of the tagging joint, in the coordinate system of the target.

Once the tagging joint is moving along the target so that it does not move with respect to the target, the more distal joints can be used to perform the finer details required by the subgoal. The programming of these distal joints will not be considered here, since it is equivalent to programming a robot to perform a task on a fixed object.

In our experimental setup, we have concentrated on reaching the goal points. Each subgoal consists of illuminating a light-sensitive diode mounted on the surface of the target for a given duration. The source of light is a laser pointer mounted on the tool plate of the robot arm. Each diode is mounted inside a tube at the focal point of a lens which closes that tube, so that the laser beam must be roughly aligned with the optical axis of the lens to trigger the electronic circuits which control the output of the diodes. Thus a goal point for the laser tool is defined by the positions at a short distance from the lens of a diode along the optical axis, and by the orientation of this axis.

8 Bringing a Tagging Joint to a Goal Point of the Target

Suppose the original trajectory of the tagging joint in location/orientation space is the vector $\vec{p_0}(t)$ (Figure 2). The goal trajectory in location/direction space is given by the vector $\vec{p_g}(t)$. At time t_0 we want the tagging joint to "launch" from its original trajectory $\vec{p_0}(t)$, and to "land" at time $t_g = t_0 + T$, on the goal trajectory $\vec{p_g}(t)$. The reaching trajectory $\vec{p_r}(t)$ should be equal to trajectory $\vec{p_0}(t)$ at time t_0 and to trajectory $\vec{p_g}(t)$ at time t_g. The operation will last for the reaching duration T. Furthermore, the first derivatives should also be equal at these times, so that the velocities change smoothly when the robot departs from its original trajectory and reaches the goal trajectory. Once a launching time t_0 and a reaching duration T are chosen, the end points of the reaching trajectory $\vec{p_r}(t)$, as well as the first derivatives of the reaching trajectory at these points are known. These boundary conditions are enough to define $\vec{p_r}(t)$ in terms of a parametric cubic spline, a curve in which all the coefficients of the six cubic polynomials of time can be calculated.

Note that the predicted motion of the target and the predicted goal point trajectories should be updated every time a new target pose is found for the target. After each of these updates, the reaching trajectory of a tagging joint should be recomputed based on the new goal trajectory, and the present joint trajectory, which may itself be a reaching trajectory started after a previous update.

9 Optimizing Reaching Trajectories

With either method of estimating reaching trajectories, one difficult problem is the preliminary choice of T, the duration of the reaching trajectory. Duration T should be chosen so that the resulting linear and angular velocities and accelerations are within the limits imposed by the robot design. Also, the reaching trajectory should not cross an obstacle or the target itself, and should not require the robot to take impossible configurations. Usually, time is the rarest commodity, and the shortest time T compatible with the above constraints should be chosen.

In our present simulations on a serial machine, the duration T is simply calculated as the time which would be necessary if the reaching trajectory followed a linear path, at a constant velocity chosen to be a safe fraction of the maximum linear velocity of the robot. Finally, we check whether this trajectory crosses robot limits or causes a collision with the target. If it does, the reaching trajectory is recalculated with a safe intermediary goal instead of the final goal point.

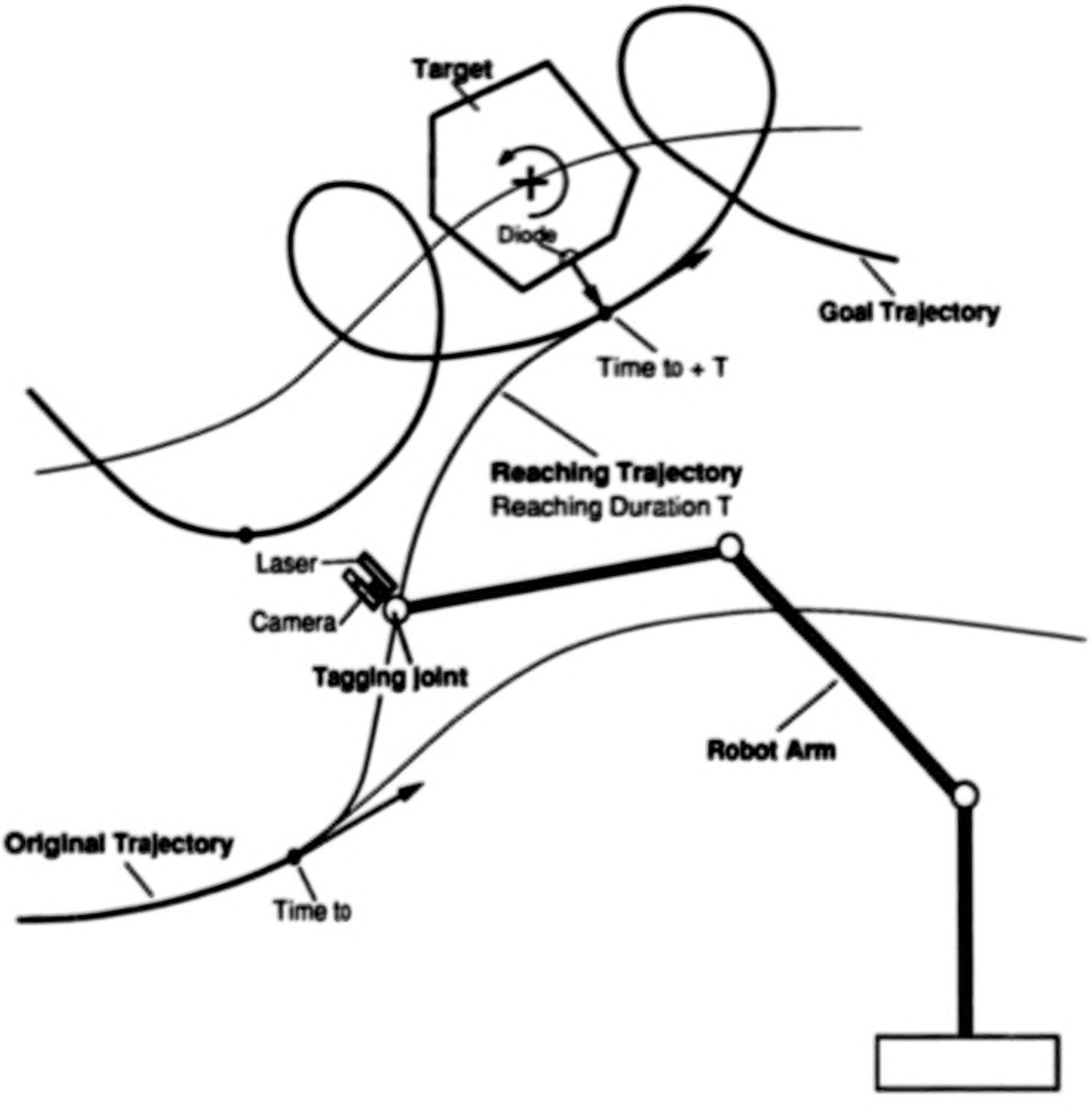

Target
Diode
Goal Trajectory
Time to + T
Reaching Trajectory
Reaching Duration T
Laser
Camera
Tagging joint
Robot Arm
Original Trajectory
Time to

Figure 2. A tagging joint takes a reaching trajectory during time T to reach a goal trajectory required to grip a handle on the target

A better optimization of the reaching trajectory would require a choice of duration T which would set the velocities and accelerations along the reaching trajectory close to the limit capabilities of the robot. This can be done by calculating the reaching trajectories for a series of durations T, finding the maximum of the second derivatives along the trajectories, and identifying the trajectory with the smallest duration T which does not require positions, velocities and accelerations beyond the robot capabilities. On the Connection Machine, this operation can be done in only a few steps. We set up a 2D array of processing cells with time as the vertical dimension. Every column of the array contains a copy of the predicted goal trajectory, with the first cell containing the position of the goal at the present time in location/direction space, the next cell the position at a time increment in the future, and so on. Every column also contains a copy of the trajectory of the tagging joint, sampled with the same time increments as the goal trajectory. The difference between columns is that they use different durations T of the reaching trajectory, increasing from one column to the next.

Each cell computes a point of the reaching trajectory for the time t corresponding to its row and for duration T corresponding to its column, and then computes estimates of appropriate derivatives at its reaching trajectory point by communicating with its neighbors in the column. The maxima of the derivatives are computed for each column. The column that has the smallest duration T and for which the maxima of the positions and derivatives do not violate robot limits is the column which contains the desired reaching trajectory. The near-term future motion of the robot should be controlled based on this selected trajectory.

10 Higher Level Planning

In a complex action we have a set of tasks A, B, C, D, each of which requires a tagging joint to move smoothly to a specific goal trajectory. In some actions the order of the tasks is not specified, and we have to choose a good order in which to carry out the tasks. "Good" order here means one which minimizes the total time spent moving between goal trajectories. Notice that this is not equivalent to a travelling salesman problem, since the time required to move from performing, say, task B to task D, depends on when we move from B to D, therefore depends on the sequence of tasks which have been performed before B.

10.1 Greedy Approach

At any given time, we can compute all the reaching trajectories to all the goal trajectories of the remaining n tasks. From among these n reaching trajectories, we can choose the one with the shortest duration, and pursue the corresponding task. This could possibly be repeated in real time after each task is accomplished, but does not guarantee the best overall sequence of tasks.

However, given that our model of the anticipated motion of the target is being updated as new information arrives, this procedure may make the most sense, in that there may be no point in computing a global optimal sequence of tasks based on a model of anticipated target motion which will not remain correct in the future.

10.2 Exhaustive Search and Dynamic Programming

Assuming, however, that our model of anticipated motion is accurate enough to allow a meaningful computation of an overall optimal sequence of tasks, one (unattractive) possibility is to compute the total time required to complete all the tasks for all possible orderings of the tasks. For n tasks this will involve computing

$$\sum_{i=1}^{n} \frac{n!}{(i-1)!}$$

best reaching trajectories.

A dynamic programming method has been developed which can precompute the best overall sequence of n tasks using computation proportional to

$$\sum_{i=1}^{n} \frac{n!}{(n-i)!(i-1)!}$$

For four tasks, for example, this approach would require computing 32 reaching trajectories rather than the 64 required for the exhaustive approach.

11 Conclusions

We have described research on robots acting in dynamic environments. We discussed the use of vision to assess the motion of objects relevant to the robot's goals, and the use of particular motion prediction and planning techniques to accomplish these goals. We described a set of experiments currently under way involving a robot arm equipped with a camera and laser operating on a single moving target object equipped with light sensors. Finally, we detailed the parallel implementation of many of the tasks involved in the accomplishment of goals in dynamic environments, including parallel image processing, parallel pose estimation, and parallel planning.

12 Acknowledgements

The support of the Defense Advanced Research Projects Agency and the U.S. Army Engineer Topographic Laboratories under Contract DACA-76-88-C-0008 is gratefully acknowledged.

REFERENCES

[1] M. Asada, Y. Fukui, and S. Tsuji, "Representing a Global Map for a Mobile Robot with Relational Local Maps from Sensory Data", International Conference on Pattern Recognition, 1988, 520-524.

[2] R.A. Brooks, "Solving the Find-Path Problem by a Good Representation of Free-Space", *IEEE Trans. Systems, Man, and Cybernetics*,13, no.3, March-April 1983, 190-197.

[3] J.L. Crowley, "Navigation for an Intelligent Mobile Robot", *International Journal of Robotics Research* ,1, March 1985, 31-41.

[4] S.J. Dickinson and L.S. Davis, "An Expert Vision System for Autonomous Land Vehicle Road Following", Computer Vision and Pattern Recognition, 1988, 826-831.

[5] B. Faverjon and P. Tournassoud, " A Local-Based Approach for Path Planning of Manipulators with a High Number of Degrees of Freedom", Proc. IEEE Robotics and Automation, 1987, 1152-1159.

[6] M. Kabuka and A.E. Arenas, "Position Verification of a Mobile Robot using Standard Pattern", IEEE Journal of Robotics and Automation, 3, 1987, 505-516.

[7] T. Lozano-Perez, "A Simple Motion-Planning Algorithm for General Robot Manipulators", *IEEE Journal of Robotics and Automation*, 3, June 1987, no.3, 224-238.

[8] H.P. Moravec, "Sensor Fusion in Certainty Grids for Mobile Robots", *AI Magazine*, 9 (2), Summer 1988, 23-104.

[9] K. Sugihara, "Some Location Properties for Robot Navigation using a Single Camera", *Computer Vision, Graphics and Image Processing*, 42, 1988, 112-129.

[10] C. Thorpe, M.H. Hebert, T. Kanade, and S.A. Shafer, "Vision and Navigation for the Carnegie-Mellon Navlab", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 10, 1988, 362-373.

[11] M.A. Turk, D.G. Morgenthaler, K.D. Gremban, and M. Marra, "VITS–A Vision System for Autonomous Land Vehicle Navigation", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 10, 1988, 342-361.

[12] K. Fujimura and H. Samet, "A Hierarchical Strategy for Path Planning among Moving Obstacles", Center for Automation Research CAR-TR-237, University of Maryland, College Park, November 1986.

[13] K. Fujimura and H. Samet, "Accessibility: A New Approach to Path Planning among Moving Obstacles", Computer Vision and Pattern Recognition, 1988, 803-807.

[14] Q. Zhu, "Structural Pyramids for Representing and Locating Moving Obstacles in Visual Guidance of Navigation", Computer Vision and Pattern Recognition, 1988, 832-837.

[15] N. Kehtanrnavaz and S. Li, "A Collision-Free navigation Scheme in the Presence of Moving Obstacles", Computer Vision and Pattern Recognition, 1988, 808-813.

[16] P.S. Schenker, R.L. French, D.B. Smith, "NASA telerobot testbed development and core technology demonstration", SPIE Conference on Space Station Automation (IV), vol.1006, Cambridge, MA, November 1988.

[17] D.H. Ballard and C.M. Brown, *Computer Vision*, Prentice-Hall, 1982.

[18] A. Rosenfeld and A.C. Kak, *Digital Picture Processing*, second edition, Academic Press, New-York, 1982.

[19] D. Harwood, M. Subbarao, H. Haklahti, H. and L.S. Davis, "A New Class of Edge Preserving Filters", *Pattern Recognition Letters*, 6, 1987, 155-162.

[20] S. Linnainmaa, D. Harwood, D. and L.S. Davis, "Pose Determination of a Three-Dimensional Object using Triangle Pairs", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 10, 1988, 634-647.

[21] K. Kanatani, "Constraints on Length and Angle", *Computer Vision, Graphics and Image Processing*, 41, 1988, 28-42.

[22] Y. Ohta, K. Maenobu and T. Sakai, "Obtaining Surface Orientation of Plane from Texels under Perspective Projection", Proc. IJCAI, 1981, 746-751.

[23] J. Aloimonos and M. Swain, "Paraperspective Projection: Between Orthography and Perspective", Center for Automation Research CAR-TR-320, University of Maryland, College Park, May 1987.

[24] D.W. Thompson and J.L. Mundy, "Three-Dimensional Model Matching from an Unconstrained Viewpoint", Proc. IEEE Robotics and Automation, 1987, 208-220.

[25] D.W. Thompson and J.L. Mundy, "Model-Directed Object Recognition on the Connection Machine", Proc. DARPA Image Understanding Workshop, 1987, 98-106.

[26] R. Horaud, "New Methods for Matching 3-D Objects with Single Perspective Views", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 9, no.3, May 1987.

APPENDIX A
Lookup tables for the pose calculation of a triangle from an image

A triangle is defined by two sides from a common vertex (the reference vertex) and the angle between them (reference angle). We assume that a standard camera rotation and a camera roll have brought the reference vertex of the image triangle to the image center and one side parallel to the X-axis of the image. Notations for the known angles and side lengths of the image and world triangle are shown in Figure A1. Three (yet unknown) numbers define the position of the world triangle in space:

1. Angles θ_1 and θ_2 of the lines P_0P_1 and P_0P_2 with respect to the Z-axis.
2. Distance R from the center of projection O to the vertex P_0 on the Z-axis. Once the angles have been obtained the calculation of R is straightforward and does not require a table.

The perspective transformation which yields the image triangle from the world triangle is approximated by a paraperspective transformation, which amounts to a sequence of two transformations, a local orthographic projection of the world triangle on a plane parallel to the image through its reference vertex, and a perspective transformation from the resulting image to the actual image (Figure A2).

Looking at Figure A2, we see that

$$\frac{f}{R} = \frac{d_1}{D_1 \sin\theta_1} \tag{1}$$

Similarly, for the image segment p_0p_2 and P_0P_2

$$\frac{f}{R} = \frac{d_2}{D_2 \sin\theta_2} \tag{2}$$

We then define the parameter s, the ratio of the two sides of the image triangle, the parameter S, the ratio of the two corresponding sides of the space triangle, and the parameter K, the ratio of s and S :

$$s = \frac{d_1}{d_2},\ S = \frac{D_1}{D_2},\ K = \frac{s}{S}$$

The parameter K is a known constant of the problem, since the dimensions of the image triangle and of the world triangle are both known. Dividing Equations 1 and 2, we find that the ratio of the sines of the angles θ_1 and θ_2 must be equal to this constant K:

$$K = \frac{\sin\theta_1}{\sin\theta_2} \tag{3}$$

The dot product of the two unit vectors $\vec{n}_1$ and $\vec{n}_1$ parallel to P_0P_1 and P_0P_2 is equal to $\cos\alpha$. These two unit vectors have components $(\sin\theta_1, 0, \cos\theta_1)$ and $(\sin\theta_2\cos\phi, \sin\theta_2\sin\phi, \cos\theta_2)$, where ϕ is the angle of p_0p_2 with the X-axis. The dot product is

$$\cos\alpha = \sin\theta_1 \sin\theta_2 \cos\phi + \cos\theta_1 \cos\theta_2 \tag{4}$$

The angles θ_1 and θ_2 are unknown. The fact that the ratios of the sines must be equal to a known constant (Equation 3) allows the elimination of one of the unknowns. The result is a second degree equation in $\sin^2\theta_1$. We define $X_1 = \sin^2\theta_1$ and find

$$\sin^2\phi X_1^2 - (K^2 - 2K\cos\alpha\cos\phi + 1)X_1 + K^2\sin^2\alpha = 0 \tag{5}$$

Equation 5 always has two positive solutions, but only the smaller solution is smaller than 1 and can be made equal to a sine. Thus we always get a single solution for $\sin\theta_1$

$$\sin\theta_1 = \left[\frac{-B - \sqrt{\Delta}}{2\sin^2\phi}\right]^{1/2} \tag{6}$$

with

$$B = -(K^2 - 2K\cos\alpha\cos\phi + 1)$$

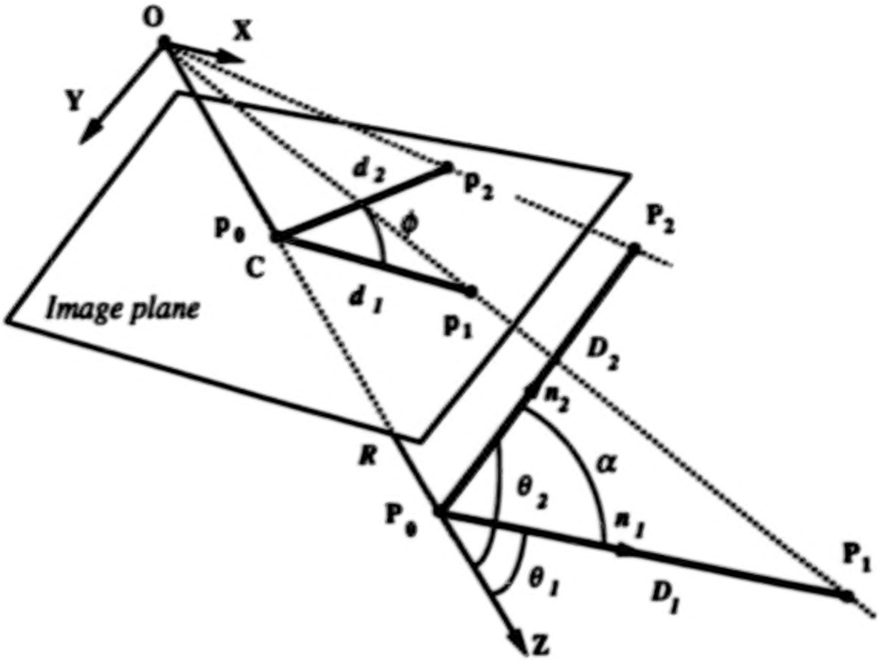

O
X
Y
d_2
p_2
P_2
p_0
C
ϕ
d_1
p_1
Image plane
D_2
n_2
α
R
θ_2
P_0
n_1
P_1
θ_1
D_1
Z

Figure A1. Image triangle with vertex at image center, and pose required for a given world triangle to match this image triangle

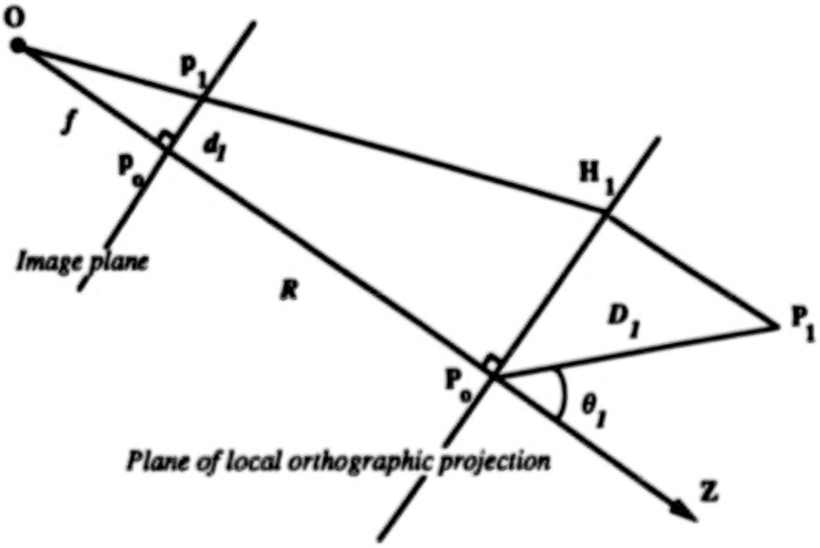

O
p_1
f
p_0
d_1
Image plane
H_1
R
D_1
P_1
P_0
θ_1
Plane of local orthographic projection
Z

Figure A2. Perspective transformation of triangle side P_0P_1 approximated by a paraperspective transformation giving image triangle side p_0p_1.

$$\Delta = B^2 - 4K^2 \sin^2\alpha \sin^2\phi$$

This single sine solution gives two solutions θ_1 and $\pi - \theta_1$, which correspond to mirror image directions of P_0P_1 with respect to the plane parallel to the image plane through P_0. The corresponding value for $\sin\theta_2$ found from Equation 3 also gives two solutions θ_2 and $\pi - \theta_2$. Note that we cannot combine the two θ_1 solutions and the two θ_2 solutions in all four ways. Indeed each of the two θ_1 solutions corresponds to cosines of opposite signs (similarly for θ_2), so that two combinations of θ_1 and θ_2 give a positive number for product $\cos\theta_1 \cos\theta_2$, whereas the other two give a negative product. To choose which two combinations out of the four are the correct ones, we rewrite Equation 4

$$\cos\theta_1 \cos\theta_2 = \cos\alpha - \sin\theta_1 \sin\theta_2 \cos\phi$$

and we check the sign of the right-hand side. Finally, the distance R of the vertex P_0 from the center of projection is computed using Equations 1 or 2. For each target triangle, a table is built with two entries, one for the image angles ϕ, the other for the ratios K between the image and world triangle side ratios. The values read can be the (θ_1, θ_2) solutions, or expressions which require the values of θ_1 and θ_2. If there are 50 characteristic triangles in a target object, 50 tables are precalculated. Each table would be of size 120×128 if image angles ϕ are quantized in 1.5 degree increments from 0 to 180 degrees, and if 128 values of K are considered. Since variations of θ_1 and θ_2 are small for K large, a log scale is used for K. The distance R, which completes the description of the pose of the triangle in space is obtained from θ_1 or θ_2, and from the ratio of a target edge length to its image edge length (Equation 1 or 2).

Appendix B
Pose calculation on the Connection Machine

To run the pose estimation algorithm described in Appendix A, the Connection Machine is used in three ways:

1. as a lookup table engine
2. as a combinatorial machine, for taking care of all the combinations of target triangles and image triangles in parallel
3. as an image processor, for calculating convolutions and finding peaks in 2D Hough transform space.

1. Lookup table engine:

We set up 2D arrays of processing cells. For the standard rotation computation, each cell represents a pixel of the image, and contains the rotation parameters required when the reference vertex of an image triangle is located at that pixel, and also contains the rotation parameters for the inverse rotation.

For the computation of the target triangle orientations, a different 2D array is precomputed for each target triangle. The vertical dimension corresponds to different reference angles of the image triangles, in equal increments; the horizontal dimension coresponds to different edge ratios of the image triangles, in log scale because variations of target triangle orientation are small when the edge ratio is large. Since we are only interested in clustering the target centers, each cell skips the computation step of finding the target triangle oriention and directly determines the coordinates of the vector joining the target triangle reference vertex to the target center.

2. Combinatorial machine:

The lookup tables previously described have been created once and for all for a given target geometry and are then used repeatedly as new images of the target are analyzed. We now describe the distribution of data and operations at run time. We again set up a 2D array of processing cells. Rows are assigned to target triangle data. The coordinates of the reference vertex in the target coordinate system and the edge ratio of each target triangle are copied to all the cells of a specific row prior to run time. Columns are assigned to image triangle data, which are refreshed after each image capture and analysis. The coordinates of the three vertices of an image triangle are copied to all the cells of a specific column. Also copied are the standard rotation and inverse rotation parameters fetched from the standard rotation lookup table. Consequently, each cell within the range of columns and rows filled up by target and image triangles contains a different combination of target triangle and image triangle. Each cell can then:

- transform the image triangle by standard rotation and swing around the optical axis, finding the new edge ratio and reference angle

- find the vector from the reference vertex to the target center from a lookup table
- find the distance of the target triangle vertex
- use previous information and the inverse rotations to find the coordinates of the target center and its image, and the rotation matrix of the target.

3. Image processor:

We set a 2D array in which each image pixel is assigned to a cell. The bin-count of a cell is incremented if the projection of the target center falls on the corresponding pixel.

The image is then smoothed. Each cell having a bin-count above a given threshold is considered a candidate cluster. These cells are numbered by decreasing bin-count. This number is broadcast back to the 2D array of image triangles and target triangle combinations which contributed to the corresponding clusters. The other cells of this array are disabled.

The 2D clustering of the images of the target center might not yield the correct target pose. Thus we also cluster the target center with respect to its Z-coordinate. We set up a 2D array with one row of cells per selected image cluster and one column per increment of the Z-coordinate. A bin-count increment for a cell of this array occurs when a cell of the image and target triangle array contains the cluster number corresponding to the row and the z-coordinate corresponding to the column. Then within each row we perform a 1D smoothing. The row which contains the highest cluster is selected, and in the image and target triangle array only the cells which contributed to this cluster are selected.

Finally, among these selected cells, the average value of the position of the target center is found, while dropping the outliers. The average rotation matrix of the target center is also calculated at this point. Our experiments with a simple polyhedra gave unambiguous results without further clustering in about one second, but experiments with more complex objects might require additional clustering with respect to the rotation matrix.

Data Management

BLANK PAGE

DEVELOPMENT OF AN INTELLIGENT INTERFACE FOR ADDING SPATIAL OBJECTS TO A KNOWLEDGE-BASED GEOGRAPHIC INFORMATION SYSTEM

William J. Campbell
National Space Science Data Center
Goddard Space Flight Center

Craig Goettsche
Science Applications Research Inc.
National Space Science Data Center
Goddard Space Flight Center

ABSTRACT

Earth Scientists lack adequate tools for quantifying complex relationships between existing data layers and studying and modeling the dynamic interactions of these data layers. There is a need for an earth systems tool to manipulate multi-layered, heterogeneous data sets that are spatially indexed, such as sensor imagery and maps, easily and intelligently in a single system. The system can access and manipulate data from multiple sensor sources, maps, and from a learned object hierarchy using an advanced knowledge-based geographical information system.

Such a prototype system, called KBGIS (Knowledge-Based Geographic Information System) was recently constructed at the University of California, Santa Barbara campus and funded by the United States Geological Survey. Although much of the system internals are well developed, the system lacks an adequate user interface, and would benefit from being able to input and output imagery data in NASA formats. The application of KBGIS would be of great benefit to current NASA earth science research as well as providing a more sophisticated understanding of the issues and required technologies in the upcoming EOS era.

This paper describes a methodology for development of an intelligent user interface and extending KBGIS to interconnect with existing NASA systems, such as imagery from the Land Analysis System (LAS), atmospheric data in Common Data Format (CDF), and visualization of complex data with the NSSDC Graphics System (NGS). This would allow NASA to quickly explore the utility of such a system, given the ability to transfer data in and out of KBGIS easily. The use and maintenance of the object hierarchies as polymorphic data types brings, to data management, a whole new set of problems and issues, few of which have been explored above the prototype level.

INTRODUCTION

It is a well known fact that the sheer volumes and complexity of scientific data being generated today require sophisticated technologies to locate, access, manipulate and display these data if they are to be of any significant scientific value. One technology that has come into its own is Geographic Information Systems (GIS) which has matured over the last twenty years such that it is now a multimillion dollar industry. GIS's allow almost any kind of data to be organized into a digital format such that any file, image or picture is transformed into a two-dimensional grid of points. For example, if a variable is an intensity function, say from a weather satellite, then this radiance value to percentage reflectance can be assigned a number expressing that intensity. Once this is accomplished all data now possess spatial singularities in that they may be referenced to a common geographical base. Usually what is done with geobased data is to reduce or enlarge and rectify the spatial attributes to one scale and projection. Now that the database is established in a commonly organized format, decision rules permit assignment of the relevant information from one or more classes that happen to fall within a given spatial sector or cell. This allows for a straightforward process for the user to compare data sets, produce new overlay combinations, assess the influence or interaction of different variables or map separates and provide input into models the user is trying to construct. This in essence is the "heart" of GIS data management. [CAM81]

Distinction between GIS and a Database Management System (DBMS)

"A Data Base Management System (DBMS) is a set of programs used for the manipulation and retrieval of logically related files containing data and structural information; it allows analysis to be used in some decision-making process. A GIS is a geo-referenced system for the specification, acquisition, storage, retrieval, and manipulation of data. These data may be related to a place; that is, the data elements link the data to a location identifier (e.g., Goddard Space Flight Center), whereas a DBMS does not require the location distinction". [CAM81] Figure1 is a graphical representation of a typical GIS data handling approach. As is apparent from Figure 1, a GIS should have at a minimum, the following capabilities:

- Data Input
- Encoding Procedures
- Data Management
- Manipulative Operations
- Output Products

Although GIS technology has come a long way since the mid 1960's, it still suffers from a variety of shortcomings such as a standardized, uniform interface, an accurate data capture and encoding mechanism, full integration with other related technologies, (e.g., image processing relational data models, graphics etc.). However, several universities and private vendors are taking a variety of approaches to address these shortcomings, one of which is described in the next section.

Artificial Intelligence and KBGIS

The logical fusion of Artificial Intelligence (AI) technologies to the complex, multifaceted capabilities of GIS's is an obvious one. Natural language query processors, expert systems, knowledge acquisition tools and artificial neural nets are but a few techniques that hold great promise for enhancing existing GIS shortcomings. One approach taken by the University of California Santa Barbara (UCSB) has been the establishment of a project to develop a Knowledge-Based Geographic Information

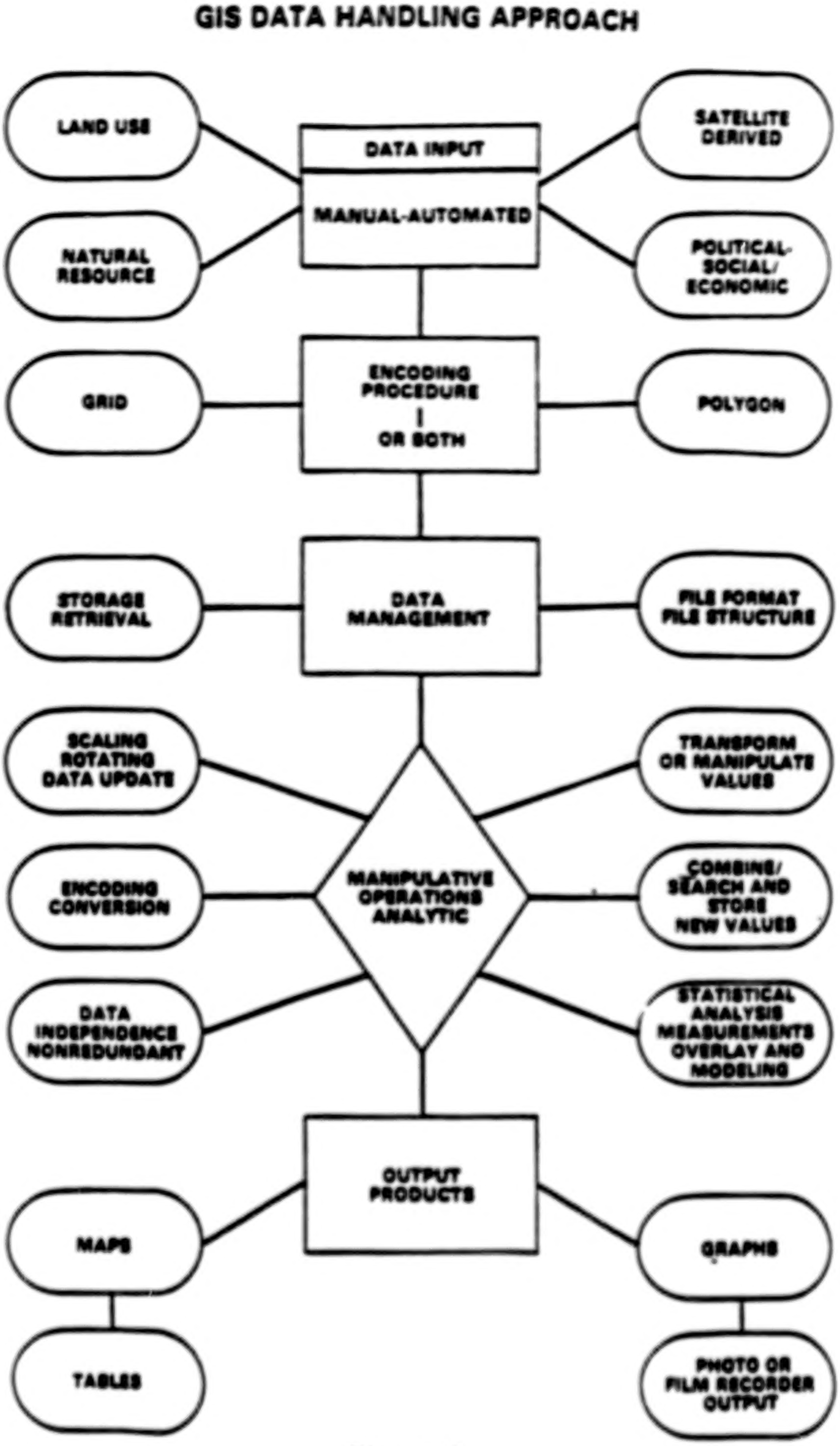

GIS DATA HANDLING APPROACH
LAND USE
DATA INPUT
MANUAL-AUTOMATED
SATELLITE DERIVED
NATURAL RESOURCE
POLITICAL-SOCIAL/ ECONOMIC
GRID
ENCODING PROCEDURE I OR BOTH
POLYGON
STORAGE RETRIEVAL
DATA MANAGEMENT
FILE FORMAT FILE STRUCTURE
SCALING ROTATING DATA UPDATE
TRANSFORM OR MANIPULATE VALUES
ENCODING CONVERSION
MANIPULATIVE OPERATIONS ANALYTIC
COMBINE/ SEARCH AND STORE NEW VALUES
DATA INDEPENDENCE NONREDUNDANT
STATISTICAL ANALYSIS MEASUREMENTS OVERLAY AND MODELING
OUTPUT PRODUCTS
MAPS
GRAPHS
TABLES
PHOTO OR FILM RECORDER OUTPUT

Figure 1.

BEST COPY AVAILABLE

System (KBGIS). Funded and supported by the United States Geological Survey and NASA, KBGIS is being developed to accommodate high-level expert system rules and heuristics, efficient spatial data search techniques through the use of new data structures and develop a learning capability by interaction from a user. The basic difference between a knowledge-based GIS from traditional GIS's or for that matter DBMS's is that a Knowledge-based GIS embodies the user's point of view about the data layers and the manipulation procedures that can be implemented on those layers as well as providing standard GIS capabilities.

The current capabilities of KBGIS include a KBGIS specific interface, which allows a user to query the system to find sets of spatial locations which the user described as well as to search for specific spatial objects that exist at a specifically predetermined location, a knowledge base which stores definitions and other useful information about objects in the system, and a learning procedure that is designed to reduce query search time. The system was developed on a VAX 11/750 using VMS and programmed in Common Lisp, C, and Pascal. Ongoing research at UCSB includes the improvement in file structures and indexing methods, integration of remotely sensed data, incorporation of useful image processing techniques and the extension of methodologies to improve vector data input. [ALB88]

National Space Science Data Center (NSSDC) KBGIS Development

KBGIS maintains each data layer in an indexed quad tree, and has several spatial operators to access and combine layers into a new layer. Knowledge about each layer is stored in a frame-based language, so that queries about the entire layer can be answered quickly. The quad tree allows a natural hierarchical pyramid of resolution that can be exploited during spatial searching to quickly eliminate quads that do not have the desired objects. As more is learned about a layer, that information can be added to the frame describing that layer, thus making the system smarter with use. This capability is a significant advancement over traditional GIS's. In addition to the spatial data, KBGIS has a frame-based layer of objects either extracted from the data, or inserted into the object knowledge base by the user. The objects may have a location attribute that points to where the object was discovered or derived, or they may be abstract types that help form hierarchies of the objects. Objects are represented in the Spatial Object Language (SOL), which allows complex, nested expressions of ANDS, ORS, NOTS, and other operators to compose objects of multiple parts. Also, to allow new spatial operators to be added easily to the system, KBGIS has a function knowledge base that allows users to add new operators to the query language. The operators can access and manipulate the data layers to perform discipline specific operations. [SMI87]

Through an informal memorandum of understanding with the USGS, the National Space Science Data Center (NSSDC) is currently developing a UNIX-based version of KBGIS implemented on a Sun 3/260 workstation. In addition, we are extending KBGIS to interconnect with existing NASA systems, such as imagery from the Land Analysis System (LAS), atmospheric data in Common Data Format (CDF), and visualization of complex data with the NSSDC Graphics System (NGS). These capabilities allow the utility of easily transferring data in and out of KBGIS. Figure 2 illustrates the specific work described in the next section.

BLANK PAGE

CONCEPT OVERVIEW

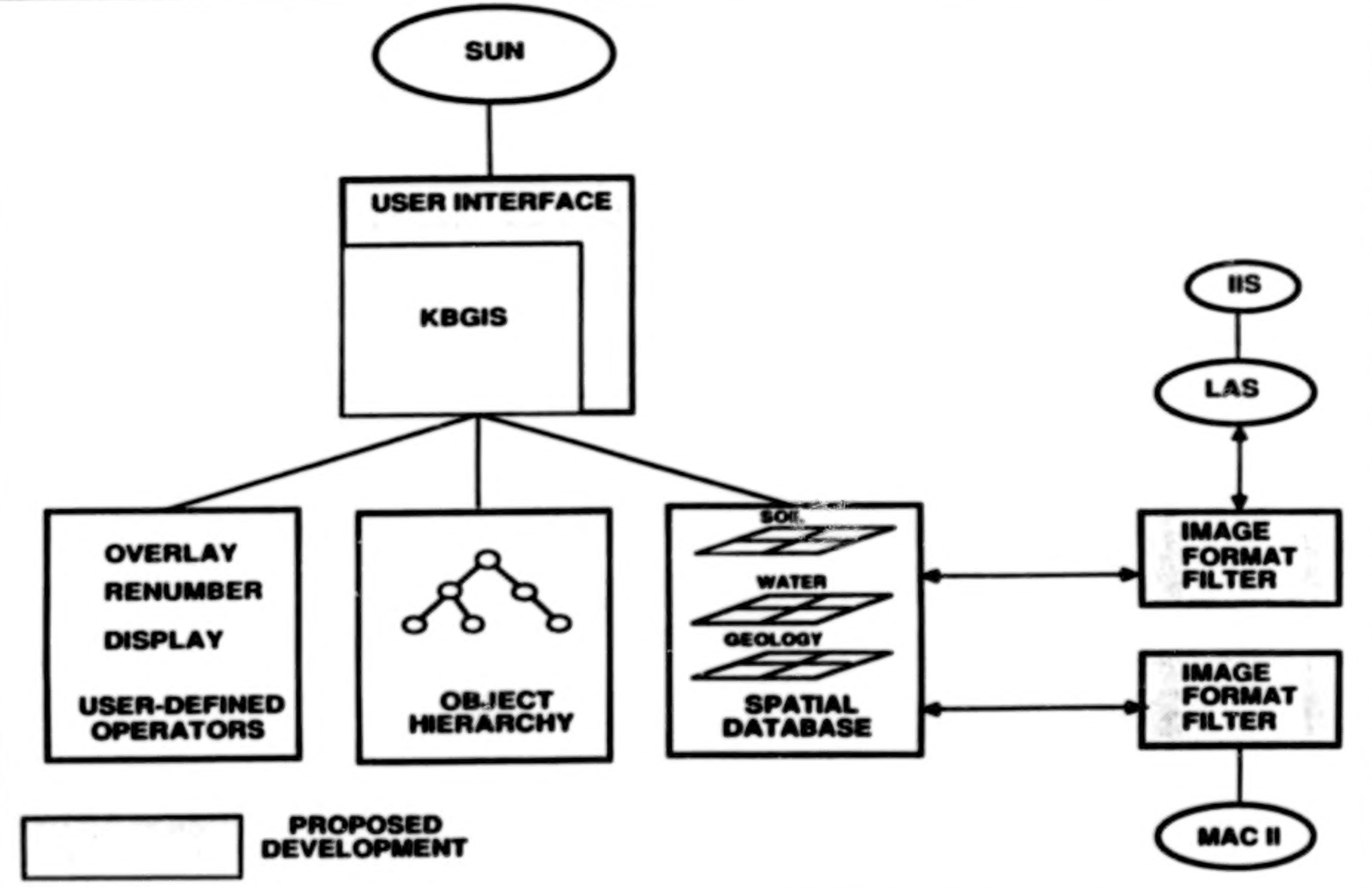

SUN
USER INTERFACE
KBGIS
OVERLAY
RENUMBER
DISPLAY
USER-DEFINED OPERATORS
OBJECT HIERARCHY
SOIL
WATER
GEOLOGY
SPATIAL DATABASE
IIS
LAS
IMAGE FORMAT FILTER
IMAGE FORMAT FILTER
MAC II
PROPOSED DEVELOPMENT

Figure 2.

BEST COPY AVAILABLE

Current Status

With the recent developments in computer technology, it is becoming unreasonable to demand that an application user be required to use a proprietary software system on a specific hardware platform. The same rules apply to GIS software. The evolving software standards, which have recently emerged (e.g., UNIX, X-Windows, PHIGS), allow software developers to design applications that can be transported to various hardware environments much more easily. The eventual goal of these software standards is to create applications that not only run on a variety of equipment, but also make the particular hardware environment transparent to the the user.

The immediate goal of the research on KBGIS at the NSSDC is to transform KBGIS into a portable system. As previously mentioned, the initial version of KBGIS was developed on a VAX. All of the low level input/output routines, which are required for all primitive object searches of the spatial location data base, are written with system specific (RMS) I/O functions. These system specific functions not only limit the software to be tied to the VAX, but force the data that has been stored using these functions to reside on the VAX as well. The first step of this transformation has been to rewrite the spatial location data base I/O routines to use a more standard UNIX I/O library. With the continuing increase in both CPU speeds and disk accessing speeds, there should no longer be a need to use system specific I/O routines. The indexing strategies that are offered with the VAX I/O routines have been rewritten with routines from the UNIX I/O library and provide the same functionality although at somewhat slower speeds. However, once the data exists in a format created by the UNIX I/O functions, it can be easily transported to other systems either directly, if both systems share the same file structure, or by transferring the data via some remote connection routines such as Remote Procedure Calls (RPC) and eXternal Data Representation (XDR). RPC and XDR are two examples of tools that allows software running on two different hardware environments to communicate and transfer data across a network. [SUN86]

The original KBGIS system only supported one format, the Earth Resource Data Analysis System (ERDAS) for import and export into the system. To promote greater use in the NASA community, several additional formats widely used on various NASA projects have been added to the system. The Land Analysis System (LAS) is an image processing system developed by NASA. LAS provides the capability to import, analyze and manipulate images. One of the main features of LAS that directly relates to KBGIS is the ability to register geometrically one image to some map (or ground) location or multiple images from various instruments to each other. With the ability to directly import LAS images into KBGIS, a vast amount of spatially distributed data becomes available to create new data layers that can be immediately preprocessed and analyzed by KBGIS. [NAS87]

The Common Data Format (CDF) is another format gaining support with increased use in the NASA community as well as other agencies. CDF is a software package written at NSSDC which consists of a library of routines that form a data-independent abstraction for multidimensional data structures. [TRE87] By adding the ability to export query results from KBGIS into CDF format, many of the applications written using CDF have also become available. One such application is the NSSDC Graphics System (NGS). NGS is another application developed at NSSDC to visualize scientific data. Many of the functions of NGS have been developed to support the visualizing of geographic data. For example, geographically stored data is ingested into NGS from CDF format and stored in an "oct-tree" data structure for quick retrieval of a particular geographic region. [SAM84] NGS also supports four-dimensional plot types with the use of a color lookup table and a mapping capability that supports over 20 general projections.

Although much research has been directed towards object hierarchy and spatial search, [SMI86] minimal effort has gone into developing the user interface, making it difficult to use the original system. For instance, the interface to KBGIS did not have knowledge of the domain's high-level concepts, but instead forced the user to render his query in terms of the low-level ordering of the spatial object knowledge base. With the recent emergence of the X-Window System and, more specifically, the X-toolkit (Xtk) as a standard library for window management, creating user interfaces which can be ported to various hardware platforms has become increasingly easier. With the use of the standard X protocol, applications can retain the same look and feel across different environments. As other layers of user interface tools become available, such as Open Look, development of these types of applications will become even simpler. [SCH86]

With the use of Xtk, objects known as "widgets" are used to create various user interface elements such as menus, forms, scroll bars and item selection buttons. An additional layer, using this core-widget set, is built onto KBGIS to provide a user-friendly interface into the system. A user is able to form spatial queries without any knowledge of the spatial object language (SOL), which KBGIS uses to represent objects. The system interacts with the user in terms of the domain knowledge which the system uses to directly access the spatial object knowledge base. The new software layer then transforms the user's input into a standard SOL query which then directs the system's search. As the user interacts with the system, the spatial knowledge base is dynamically maintained so that the system retains the modified or newly created objects defined by the user. This removes the burden from the user of having to render the queries in terms of pixel properties (e.g. agriculture, residential) or relational properties (e.g. distance), and permits him to interact with KBGIS at a higher level. This increases the ease of using the system, and decreases the amount of time required for learning KBGIS.

Future Directions

There are several areas of research that could be undertaken to enhance KBGIS.

A) A natural language front end could be added to KBGIS to give a user a more descriptive way of querying the spatial object knowledge base.

B) The current learning system in KBGIS could be enhanced by adding learning by example thus making it more autonomous.

C) The display graphics of the current system is limited. To bring KBGIS up to the functionality of other GIS products on the market, several display capabilities could be added.

1) Add cartographic capabilities for base maps and hard copies
2) Add ability for image overlays

D) Enhance KBGIS with a more intelligent and robust inference engine. This will be required as the system evolves and becomes a more general purpose system.

Summary and Conclusions

Work at the USGS, UCSB and NASA continues to enhance the functionality and capability of KBGIS. Currently the system can respond to various types of spatial queries of large complex, heterogeneous data in a multilayered environment. Work at UCSB is focused on increasing the efficiency of spatial search using heuristics as well as on a more friendly interface. Our work at Goddard is primarily focused on providing a methodology for importing external data sets, primarily enhanced imagery, in an easy fashion as well as concentrating on a standardized, generic graphics interface on a Sun workstation. Once this work is completed, we will evaluate the feasibility of linking and customizing a natural language query processor [NLQP] and a high level expert system that will deal with the pragmatic translation between the NLQP and KBGIS. We will also determine what role artificial neural networks can play in automatically characterizing external data into a form that can be readily utilized by KBGIS.

Acknowledgement

The authors would like to thank Scott Wattawa of the National Space Science Data Center for his thoughts, ideas and expertise that are presented in the concept formulation and interface design section of this paper.

References

[ALB88] Albert, T. M., "Knowledge-Based Geographic Information Systems (KBGIS) New Analytic and Data Management Tools", Journal of Mathematical Geology, 1988.

[CAM81] Campbell, W. J., "Geographic Information Systems - A Training Module", Chapter 7 in N. M. Short., Landsat Tutorial Workbook, NASA/GSFC, May 1981.

[NAS87] NASA, "LAS User's Manual - Version 4.0", GSFC/NASA, September 1987.

[SAM84] Samet, H., "The Quadtree and Related Hierarchical Data Structures", ACM Computing Surveys, 16, 2, 187-260, June 1984.

[SCH86] Scheifler, R., and J. Gettys, "The X Window System", ACM Transactions on Graphics, 5, 2, 79-109, April 1986.

[SMI86] Smith, T., D. Peuquet, S. Menon and P. Agarwal, "KBGIS-II: A knowledge based geographic information system.", Technical Report TRCS 86-13, Department of Computer Science, University of California, Santa Barbara, May 1986.

[SMI87] Smith, T.R., S. Menon, J. Star and J. Estes, "Requirements and Principles for the Implementation and Construction of Large-Scale Geographic Information Systems", Int. J. Geographical Information Systems, 1987. Vol 1. No. 1, 13-31.

[SUN86] Sun Microsystems, "Networking on the Sun Workstation", Sun Microsystems, Mountain View, Calif., 1986.

[TRE87] Treinish, L., and M. Gough, "A Software Package for the Data-Independent Storage of Multi-Dimensional Data", Eos Transactions American Geophysical Union, 68, 633-635, 1987.

BLANK PAGE

THE UTILIZATION OF NEURAL NETS IN POPULATING AN OBJECT-ORIENTED DATABASE

William J. Campbell
National Space Science Data Center
Goddard Space Flight Center
Greenbelt, Maryland 20771

Scott E. Hill and Robert F. Cromp
Science Applications Research Inc.
National Space Science Data Center
Goddard Space Flight Center
Greenbelt, Maryland 20771

ABSTRACT

Existing NASA supported scientific data bases are usually developed and managed by a team of database administrators whose main concern is the efficiency of the data bases in terms of normalization and data search constructs. The populating of the data base is usually done in a manual fashion by row and column as the data becomes available, and the data dictionary is usually defined by the same team (at times with little input from the end science user). This process is tedious, error prone and self-limiting in terms of what can be described in a relational Data Base Management System (DBMS). The next generation Earth remote sensing platforms (i.e., Earth Observation System, EOS), will be capable of generating data at a rate of over 300 Mbs per second from a suite of instruments designed for different applications. What is needed is an innovative approach that creates object-oriented databases that segment, characterize, catalog and are manageable in a domain-specific context and whose contents are available interactively and in near-real-time to the user community. This paper describes work in progress that utilizes an artificial neural net approach to characterize satellite imagery of undefined objects into high-level data objects. The characterized data is then dynamically allocated to an object-oriented data base where it can be reviewed and accessed by a user. The definition, development, and evolution of the overall data system model are steps in the creation of an application-driven knowledge-based scientific information system.

Introduction

One of the most significant technical issues that NASA must address and resolve is the problem of managing the enormous amounts of scientific and engineering data that will be generated by the next generation of remote sensing systems, such as the Hubble Space Telescope (HST) and the Earth Observation System (EOS). The amount of data these sensors are expected to produce will be orders of magnitude greater than NASA has ever experienced. Consequently new solutions must be developed for managing, accessing and automatically inputting the data into a database in some expressive fashion that will provide a meaningful understanding and effective utilization of this data in a multidisciplinary environment.

Presently, scientific data provided by satellites and other sources (i.e., in situ measurements) are processed, cataloged, and archived according to narrow-mission or project-specific requirements with little regard to the semantics of the overall research. Scientists therefore lack knowledge of or access to potentially valuable data outside of their own field and usually access to this data is long after the actual generation of that data. What is needed is a methodology that will extract and characterize a processed data stream from a remote sensing instrument, and automatically augment appropriate data catalogs for remote browsing, at a high level of abstraction, that would be of interest to NASA's scientific community.

Concept

The concept is for the system to intercept a data stream from a remote sensing instrument and pass the data through a series of artificial neural networks that have been knowledge engineered or tuned to specifically identify and characterize data objects at a high level of abstraction using the appropriate domain-specific program. These networks or characterization agents will be controlled by a knowledge-based planner and controller that directs the identification and abstraction of objects determined to be of interest to the scientific community. These networks will run in parallel and will be activated as appropriate (predetermined in the tuning process) for the given defined context, for example, for a specific instrument or data stream. Initial passes of these networks characterize data objects at a high level, determined by a threshold level of confidence of a given object to be that object. The information obtained would automatically be inserted into an object-oriented database which builds, indexes, and maintains sets of elements and allows a user to retrieve these data as individual items or as any aggregate of objects. Once a remotely sensed data set has undergone some level of preprocessing (e.g., decompression, radiance values generated, etc.) then additional ephemeris information such as date, time and sensor ID can be added to the parent object or any subdivision of objects.

This data, along with associated meta data and data set identification will then be sent to an appropriate archive. A reference data frame that is specific to a particular domain (science or sensor specific) will then created within the context of the domain world model of the information management system. An important point concerning this process is that much of the information required to catalog and characterize the data set will already be in the knowledge base as a consequence of a priori knowledge acquisition from both the ephemeris information specific to a given sensor as well as the science information a particular sensor is designed to capture [CAM88].

Using the above approach, raw science and engineering data can be efficiently processed and stored using meaningful representations that are more suitable to a user's reasoning. The definition, development, and evolution of the meta frames, agents and overall data system model are the first steps in the evolution of an application-driven knowledge base.

Design Considerations

The design of a data cataloging and characterization system is predicated on having preexisting knowledge of the domain, the sensor devices and the interpretation of their measurements. It is fruitless to identify, store, and manipulate data if there are no guidelines that differentiate between good and bad observations, or if the integrity of the database cannot be guaranteed.

The suggested design melds subsymbolic processing by a neural network with high-level symbolic processing controlled by an expert system. This design requires a research and development effort in the following areas:

1. Architecture of a neural network which can characterize the pixels in a remotely sensed image based upon the satellite's primary bands.

2. Effective training procedures of a neural net to maximize its performance while minimizing the amount of required computer CPU time.

3. Combination of the technologies of neural network computing and expert systems.

4. Categorization of large data sets in near real time by using an associative memory model as defined by a neural network.

5. Use of an expert system that uses contextual information, such as time of year and location of image, to judge and refine the output of the neural network.

6. Use of an expert system to instantiate an object so that its representation is suitable for a database. This requires a mapping of the characterization of the image data (represented by a subsymbolic collection of pixels) to an object (represented by a symbolic collection of attribute-value pairings).

Approach

Initial research into steps 1 and 2 of the just outlined research plan will now be presented. First, we will introduce back-propagation, the type of neural network believed best suited to this task. A methodology for and the results of several experiments will be described. The conclusion will be drawn from those experiments that this style of computation appears quite favorable for the categorization of LANDSAT-4 images.

In the past few years, a style of computation termed neural networks has become popular. Perhaps the most successful type of neural network has been back-propagation [RUM86], a supervised learning procedure for training layered networks of neuron-like nodes. A layer of nodes are those nodes which are similarly connected to other layers in a network. Back-propagation networks have an input layer, an output layer, and from one to many intermediate, hidden layers. Connections are unidirectional links between two nodes (called the "from" and "to" nodes); each connection has an associated value called "weight." Each node has an activation value which is a function of the activation values of the nodes connected to it and the weights associated with those connections . There is some flexibility in the form of that activation function, the one chosen for this study is

$$o_j = \frac{1.0}{1.0 + e^{-\left(\sum_i w_{ij}\, o_i + \theta_j\right)}} \tag{1}$$

where

w_{ij} is the weight of the connection linking the i^{th} node to the j^{th} node,

θ_j is the threshold value for the jth node, and

o_i, o_j are the activation values of the i^{th}, j^{th} nodes,

and the summation is over all the "from" nodes i connected to the j^{th} node.

The activation value of the input nodes is set by the user as the desired input pattern. That activity then propagates forward, layer by layer, through the network as dictated by the network connectivity. The threshold value for a node acts as a weight from a node with a constant activation value of unity.

During learning, weights are adjusted so as to minimize a measure of the difference between the actual values of the output nodes (the output vector) and the desired values of the output nodes (the target vector) when the network is presented with the input vector to the input nodes and that activity is propagated forward. To do so requires a training set of input vectors and their associated target vectors. Training of the network proceeds in a series of two stage events: first, an input vector is presented to the input nodes and activation is fed forward through the network to produce an output vector. This output vector is compared with the desired target vector and the difference between the two vectors, the error vector, is computed. The measure of error used in this study is

$$E_p = \frac{1}{2}\sum_j (o_{pj} - t_{pj})^2 \tag{2}$$

where

E_p is the error after the p^{th} training pattern,

t_{pj} is the target activation value for the j^{th} node in the output layer in the p^{th} training pattern, and

o_{pj} is the activation value of the j^{th} node after presenting the input vector in the p^{th} training pattern and propagating activity forward.

Given these equations of activity propagation and error measurement, the derivative of the error for a unit with respect to the weights connected to that unit can be recursively calculated. Those derivatives are used to change the weights of the connections between the nodes in the network so as to reduce E upon subsequent presentation of the input vector. By repeating this forward propagation of activity and backward propagation of weight changes for each member of the training set, the connection weights slowly change to a configuration that, upon presentation of each member of the training set of vectors, produce in the output nodes, the target vector which is the correct interpretation for the current input vector. Training of these networks can require a long time, with many presentations of each member of the training set. However, once a neural network is trained, the weights can be implemented in real time software/hardware systems [FOG88]. Complex preprocessing stages can negate this advantage of the neural network approach, however.

The network trains in a series of epochs. Each epoch consisted of the presentation of one input vector in the training set from each category. Subsequent epochs use successive pixels from each category. In this way, if there are 85 pixels with type 2 in the training set, then each of those 85 pixels will be presented once to the network every 85 epochs. An alternative would be to sequentially present each pixel in the training set irrespective of category. The latter method would train pixels in the same ratio as their occurrence in the training set. Although this might be favorable to the former method in terms of the overall percentage correctly classified, it does so at the expense of less well represented categories. The method adopted trains on category type 4 as often as on category type 9.

Both damping and momentum factors are used. The damping factor premultiplies the specified weight change for a connection. The damping factor is the product of a network damping factor, 0.5 in these experiments, and the inverse of the fanin of the "to" unit for the current connection. The fanin of a unit is the number of connections which converge to the unit. The damping factor for a connection is multiplied by the weight change for that connection indicated by the current pixel presentation. The momentum factor premultiplies the accumulated weight change indicated by the previous epoch before accruing the weight changes indicated by the current epoch. The effect of the momentum factor is to smooth out the change in weights between presentations of training pixels.

In this research, values for the first four spectral bands from a LANDSAT-4 Thematic Mapper image are used as input to a back-propagation neural network. The bandpass for those spectral bands are 0.45-0.52 υm, 0.52-0.60 υm, 0.63-0.69 υm, and 0.76-0.90 υm, respectively. Each picture element (pixel) in the image is representative of a 30 x 30 meter area and quantized to 256 levels. The LANDSAT-4 imagery of the region was obtained in July, 1982 [TIL89]. The network is trained to associate the spectral data of each pixel with one of seventeen possible land cover or land use categories.

The network is trained to associate the spectral data from a pixel with the land cover or land use category for that pixel. There are a total of 21,273 pixels of valid ground truth provided; each is encoded as a one byte integer ranging from 1 to 17 (see Table 1). These pixels are contained within a 151 x 151 pixel region within the area designated as subregion 1 in a study by Williams, et al. [WIL84]. This area is about 25 miles SSE of Washington D. C. The land use and land cover data (ground truths) for the region were obtained by photo interpretation of color infrared aerial photography (1:40,000) that was collected over the area on July 13, 1982 and verified by subsequent field visits in October, 1982 [WIL84]. A 15 meter minimum mapping unit criterion was used. However, as that study states, "in the case of agricultural fields, the minimum mapping unit was utilized only to separate one field from another; no attempt was made to delineate within-field variability." Land cover categories were substituted for land use categories in "situations where the land cover components of the categories (e.g. the roofs, lawns, trees, and concrete/asphalt areas of a residential neighborhood) occupied areas with spatial dimensions approximately equal to or smaller than the 15-m minimum mapping unit." Notice that types 14, 15, and 16 are land use categories, while the remainder are land cover categories. Approximately 50 pixels at various points within the image have no ground truth specified. The neural networks neither train nor test on those pixels.

As far as the network is concerned, the ground truth label for each pixel is assumed to be correct. Inaccuracies in the ground truth label with respect to a hypothetical true category hinder the ability of the network to learn the relevant features of each of the possible categories. For example, if some of the (true) water pixels were originally miscategorized as (ground truth) conifer trees, then the network will see two radically different profiles of (ground truth) conifer trees the first being those (true) conifer tree pixels correctly categorized as (ground truth) conifer and the second, those (true) water pixels miscategorized as (ground truth) conifer. The network might end up learning that conifer can look like (true) conifer or look like (true) water, in which case the network would tend to miscategorize all (true) water pixels as conifer. Even if the network was able to generalize all categories correctly, categorizing correctly even pixels whose ground truth label was incorrect, those latter pixels would still show up as incorrectly categorized in the overall performance statistics that are gathered. In the example just given, if the network correctly classified (real) water pixels as water, then the performance statistics would necessarily mark as mistakes those (real) water pixels incorrectly labeled as (ground truth) conifer.

A subset of the available data is used for training; the remainder of the data is used to test the network. Pixels which are bounded on all four sides with pixels having the same ground truth category as the center pixel are said to satisfy the non-boundary criterion (NB). Two training sets are defined: the first consisting of all pixels in the top half of the image which satisfy the NB criterion (termed TRAIN1); the second is all pixels in the top half of the image, regardless of the ground truth of their neighbors (termed TRAIN2). Therefore, the TRAIN1 set of pixels is a subset of the TRAIN2 set. The NB criterion eliminates from the training set those pixels forming the borders between categories. Because ground truth labeling must categorize pixels into only one type, border pixels are more likely to contain (in reality) multiple ground truths. In addition, this criteria will help compensate for errors in the ground truth file where pixels are shifted by one pixel or less. Of the 10,996 pixels in TRAIN2, only 4,405 pixels were accepted into training set TRAIN1 after application of this criteria. Therefore, 6,231 pixels in the top half of the image had at least one nearest neighbor in a different category. Two test sets are defined: the first consists of all pixels in the second half of the image which satisfy the same nearest neighbor criteria as the training set (termed TEST1); the second is all pixels in the second half of the image, regardless of the ground truths of their neighbors (termed TEST2). Therefore, the TEST1 set is a subset of the TEST2 set. There are 5,184 pixels in TEST1 and 10,997 pixels in TEST2 (see Table 1). By using two training sets and two test sets, we can determine whether the network has learned fundamental relationships between input data and ground truth, instead of just memorizing input-output pairs. Neural networks can learn any non-contradictory training set, given enough hidden nodes.

Table 1 lists the population of each type of ground truth in both training sets and both test sets. The image has no pixels of class 3, "standing corn". Because there are no pixels in the training sets of either class 1 or 15, "water" and "multiple family residential", respectively, pixels classed in these two categories will probably be miscategorized by a network. Since a neural network learns only those patterns on which it is trained, the classes in the training image should be representative of the whole image. An alternative of training upon pixels in the first half of the image is to train upon the first (or a random) half of the pixels in each category. Eventually, images of additional locations at different times of the year will also be used. In general, the power and robustness of a neural network system depends upon the breadth of its training sets.

There are four input nodes to the network, one for each spectral channel. Hidden nodes are completely connected to the input layer and to the output layer. The output layer uses one node for each of the 17 possible ground truths. After presentation of spectral values to the input layer, the output node, from 1 to 17, with the highest activation value indicates the network interpretation of the land use/cover category for the currently presented pixel. Called a unary encoding, one node for each ground truth type was chosen because adjacent ground truths (in the code) are not necessarily related types. If all types were mapped to one node, however, uncertainty of interpretation cannot be indicated by any activation values. For example, with unary encoding, activation values for nodes 1 and 15 of 0.5 might indicate uncertainty between type 1 and 15. If all types were mapped to one node, type 1 was mapped to activity value 0.1, and type 15 was mapped to activity value 0.9, then uncertainty between the two types would probably manifest itself with some intermediate value, signifying an unrelated type.

Table 1
Descriptions for ground truth codes and the population of these types in the TRAIN1 training set, the TRAIN2 training set, the TEST1 test set, and the TEST2 test set.

	Number of Pixels in Set				
Type	Train1	Train2	Test 1	Test2	Description of Ground Truth Type
1	0	0	0	26	water
2	85	232	11	65	agriculture - miscellaneous crops
3	0	0	0	0	corn - standing
4	106	226	76	123	corn - stubble
5	74	206	45	312	shrubland
6	367	1307	670	1844	grassland / pasture
7	20	87	11	38	soybeans
8	19	77	126	426	bare soil - cleared land
9	2,492	4607	3,649	5,376	hardwood forest, >70% of the forest component, 50-100% canopy closure
10	138	589	65	317	hardwood forest , >70% of the forest component, 10-50% canopy closure
11	334	1238	224	830	conifer forest, >50% of the forest component, 10-100% canopy closure
12	62	362	24	237	mixed wood forest, 10-100% canopy closure
13	7	247	2	139	asphalt
14	524	1503	202	654	residential - single family housing
15	0	0	7	26	residential - multiple family housing
16	23	75	3	48	industrial / commercial
17	154	240	69	176	bare soil - plowed field
Total	4,405	10,996	5,184	10,997	

The input data are scaled linearly to a 0.1-0.9 range. A less than unit distance is chosen because activations of 0.0 or 1.0 require infinite weights. Infinite weights are undesirable because they require an infinite amount of time to learn. Scaling of the input is performed over a constant range for each channel: channel 1 is scaled between 65-165, channel 2 between 26-102, channel 3 between 20-127, and channel 4 between 55-151. These ranges were chosen so that, looking at a histogram of the data for a channel, all channel values with more than 1 pixel having that value would be inclusively within the scaled range. The result is saturation of some values with loss of potentially useful information. For example, the spectral data for the first pixel (ground truth 9) in training set TRAIN2 is 68, 26, 21, and 110 for channels 1 through 4 respectively. Those values would be scaled to 0.03, 0.0, 0.009, and 0.5 for channels 1 through 4 respectively for input to the network. If the spectral data for the second channel was 20 instead of 15, the scaled input for that channel would still have been 0.0. Using an input node for each possible spectral value for each channel would eliminate the need for any rescaling. This would multiply the number of input nodes by 256 from 4 to 1,024 and the number of connections accordingly. This is currently impractical because the large numbers of nodes and connections would require too much computer time.

The final performance of each network is measured by the proportion of the test pixels assigned to the correct land use category, the overall percentage correctly characterized (PCC). The PCC for each category type is also calculated. Anderson, et al. [AND76] suggest 85% as a minimum accuracy level of classification of remote sensor data. This level of performance is not expected at this stage.

Two investigations are described in this report. First, we determine a minimal number of hidden nodes sufficient to categorize the pixels in the training image and to categorize the test pixels as well as the training pixels. There are two reasons to minimize the number of hidden nodes: the first is that computation time for both training and testing increases with the number of hidden nodes; the second reason is that networks with too many hidden nodes can learn specifics of the training set which do not extend to the test set. The latter fault of networks with large numbers of hidden nodes is a consequence of too much representation power. Small number of hidden nodes "starve" a network into discovering the most parsimonious descriptions of regularities in the training set. Generally, simple generalizations extend to a test set more than complex generalizations. Networks with 1, 2, 3, 5, and 10 hidden nodes are trained on the TRAIN2 training set and tested on the TEST2 test set. Each network is trained 10,000 epochs with the momentum factor set to 0.5 and the network damping factor set to 0.5.

Next, we investigate the utility of the non-boundary criterion by training one 5 hidden node network on the TRAIN1 training set and another on the TRAIN2 training set. Each is trained for 10,000 epochs with the momentum factor set to 0.5 and the damping factor set to 0.5. Once training is complete, the networks are tested on both training sets and both test sets. This investigation determines the utility of limiting training to non-boundary pixels. Because those non-border points should be more separable than undifferentiated training pixels, it is expected that the PCC of non-border training pixels should be better than the PCC of undifferentiated training pixels. More importantly, because eliminating border points should reduce the amount of variance in the training set for each category, it is expected that training should be faster with non-border pixels than with undifferentiated pixels. In some cases, the network might not be able to extract the pattern in undifferentiated pixels, yet be able to do so if the training set is limited to non-boundary pixels. In that case, the PCC of the test set for networks trained on the non-boundary points would be superior to the PCC of the test set for networks trained on all pixels.

These experiments were performed on a SUN 4/280; every 1,000 epochs of training require 45, 54, 62, 78, and 118 seconds for networks with 1, 2, 3, 5, and 10 hidden nodes respectively. These times are not linearly proportional to the number of hidden nodes because of network overhead.

Preliminary Results

Table 2 shows the percentage of correctly characterized pixels for each of the different land cover/use types for networks with a varying number of hidden nodes. Naturally, all the networks miscategorize those pixels in the test set with ground truth types that were not in the training set: types 1, 3, and 15. The overall PCC for the test set is better than the overall PCC for the training set for networks with 2 or greater hidden nodes. This is because the relative frequencies of the classes in the test set is different from that of the training set.

Table 2

True positive categorization percentages of the training set and test set for each ground truth type after training for networks with varying numbers of hidden nodes. Each is trained on the TRAIN2 training set for 10,000 epochs with the momentum factor set to 0.5 and the damping factor set to 0.5.

	PCC - TRAIN2					PCC - TEST2				
Type	1	2	3	5	10	1	2	3	5	10
1	-	-	-	-	-	0	0	0	0	0
2	0	0	1	0	0	0	0	1	0	1
3	-	-	-	-	-	-	-	-	-	-
4	88	70	79	81	81	75	67	72	73	73
5	7	0	1	1	0	3	0	1	0	0
6	0	2	0	0	0	0	1	0	0	0
7	0	1	10	11	3	0	0	3	3	0
8	0	74	66	65	66	0	74	62	67	64
9	5	78	76	71	75	3	86	85	81	84
10	18	83	81	76	80	14	74	71	66	72
11	0	6	10	29	15	0	3	6	25	9
12	0	11	4	10	4	0	6	2	6	3
13	79	19	11	22	13	70	19	7	33	8
14	9	7	28	51	28	11	13	36	53	40
15	-	-	-	-	-	0	0	0	0	0
16	8	73	71	63	71	12	83	73	73	71
17	5	5	2	0	0	6	6	7	3	2
Overall	8	42	44	48	44	5	52	52	55	52

Because of the large amount of type 9 pixels, the PCC of that type has a large impact on the overall PCC. Whether a class has a high PCC or a low PCC is not dependent upon the relative frequency of those classes in the training set. This is due to the fact that the training method trains on the same number of pixels from each category during each epoch.

The 1 hidden node network can discriminate between classes 4 and 13 in the test set with PCCs of 75% and 70%, respectively. The PCCs for these classes are better than any network with more hidden nodes. Adding more hidden nodes does allow more classes to be distinguished, yet the ability to distinguish more classes comes at the expense of a decrease in the PCC of those already discriminated classes. There is no significant difference in the overall PCC beyond 1 hidden node. This suggests using many small 2 hidden node networks in unison, each network trained to discriminate between only a few categories.

Overall network performance for a network is best shown as a contingency table which shows the category into which each pixel has been placed, as a function of the correct ground truth for that pixel. A contingency table for the 5 hidden node network is shown in Table 3. The number in the *m*th row and *n*th column in the table indicates that percentage of pixels of ground truth *n* which were categorized as class *m*. The true positive characterized percentages of each category type lie on the main diagonal. This table illustrates that pixels tend to be mischaracterized in a non-random fashion. For example, although the PCC for type

Table 3

Contingency table showing the percentage of pixels in each ground truth type categorized as type 1, 2, 3, ..., 17 in test set TEST2 for a 5 hidden node network after training for 10,000 epochs on training set TRAIN2. The columns, one for each ground truth, sum to 100%. There is one row for each network categorization possibility. For example, the entry 19 in row 4 and column 5 indicates that 19% of the ground truth type 5 pixels were mischaracterized as type 4. The true positive categorization percentages are along the main diagonal and shown in bold print.

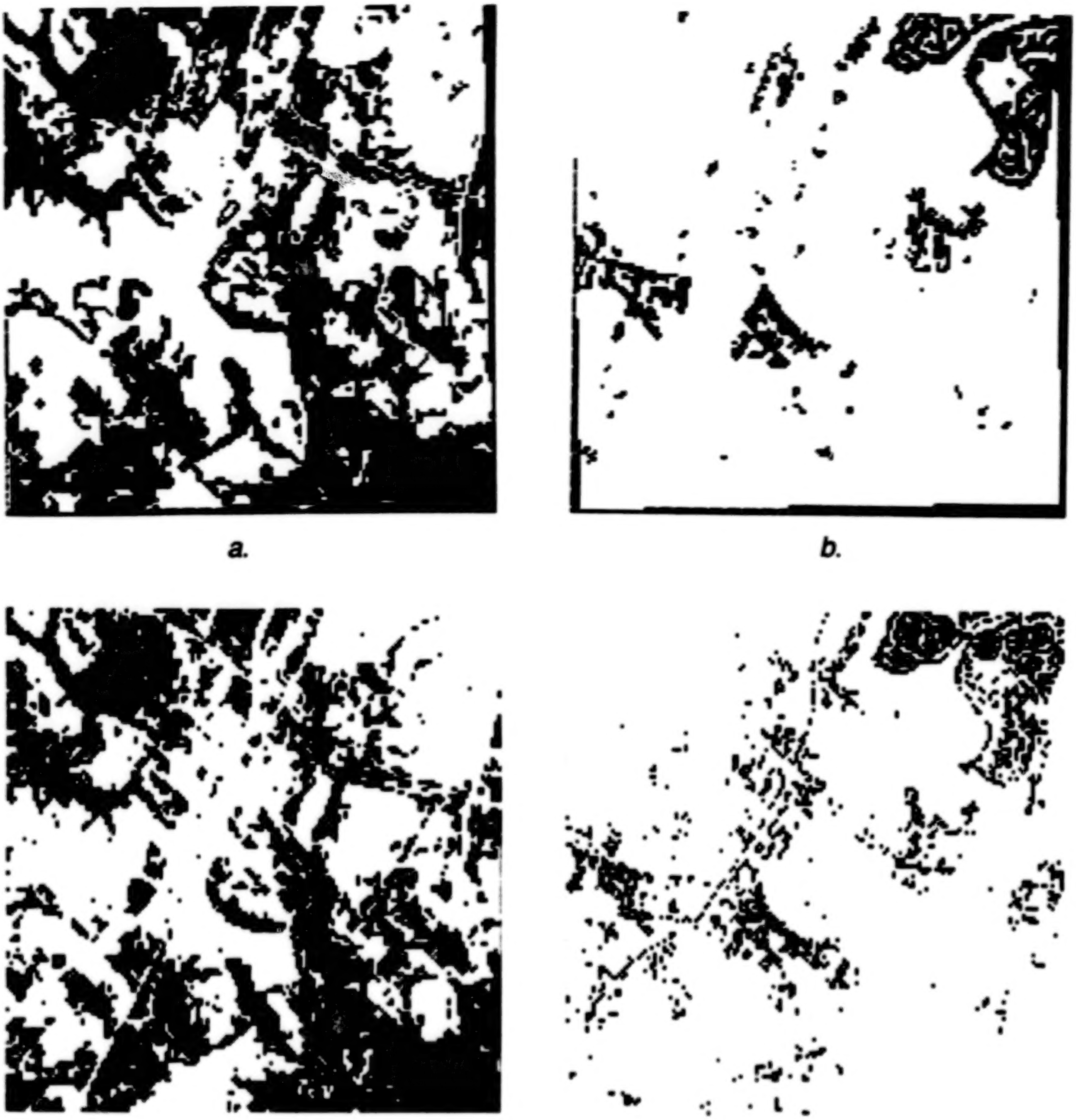
	Ground Truths																
Categorized	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	**0**	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	**0**		0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	34		**73**	19	24	5	5	3	0	1	18	2	8	0	0	74
5	0	0		0	**0**	2	0	1	0	0	0	0	0	0	0	0	0
6	0	0		0	0	**0**	0	0	0	0	0	0	0	0	0	0	0
7	0	4		1	1	7	**3**	5	0	0	0	5	3	6	0	0	3
8	39	3		15	1	5	5	**67**	1	0	1	1	0	1	35	12	0
9	0	6		0	60	13	34	1	**81**	20	27	41	0	2	0	0	17
10	14	10		2	6	3	13	1	4	**66**	39	5	5	9	0	0	0
11	0	13		3	7	8	24	0	8	11	**25**	18	1	4	0	0	0
12	0	6		3	1	11	11	1	1	1	2	**6**	2	5	0	0	1
13	4	4		0	1	5	0	2	0	1	1	0	**33**	7	0	0	1
14	4	17		1	3	17	0	8	1	1	2	6	20	**53**	27	15	2
15	0	0		0	0	0	0	0	0	0	0	0	0	0	**0**	0	0
16	39	1		0	0	2	0	6	0	0	0	0	34	4	38	**73**	0
17	0	1		2	1	3	5	3	1	0	0	1	0	0	0	0	**3**

17, bare soil - plowed fields is only 3%, most of those pixels are classified as type 4 - corn stubble. Such miscategorization is reasonable, considering the close relationship between plowed fields and corn stubble. Recall that the process of photointerpretation to determine the ground truths did not consider variability within agricultural fields.

As examples of pictures of two network types, one categorized well and another not so well, Figure 1 shows pictures of the ground truths and the network categorizations for types 9 and 14, dense hardwood forest and single-family residential, respectively. The 5 hidden node network was used to categorize all the pixels in the image after training on the TRAIN2 training set. The general pattern of the residential area, type 14, is evident in the network categorization of that type (1*d*) despite the network having only a 53% PCC in the test region TEST2 for that type. Notice that a road like line extending from the top of the image to the lower left is categorized as residential. Not shown in this figure is that the network also categorizes other pixels along this same line as asphalt. Looking at the contingency table for this network (Table 3) one can see that 20% of the asphalt (type 13) pixels were miscategorized as residential (type 14). This is possibly explained by the network becoming confused by the close relationship between roads and residential areas in the apparent subdivision in the upper right of the image which is part of the training region.

The second investigation looked into the utility of limiting training to non-boundary pixels (see Table 4). Several observations about these results can be made. First, the PCC of the training pixels are higher for the network trained on the NB pixels, TRAIN1, than for the network trained on non-distinguished pixels, TRAIN2: overall PCC of 66% vs 48%, respectively. Therefore, during training it is easier to judge that the network training on the TRAIN1 set is learning the training set than it is to judge that the network training on the TRAIN2 set is learning the training set. Yet, when the network trained on the TRAIN2 set is subsequently tested on the TRAIN1 set, the overall PCC of 65% is comparable to the PCC for the network trained on the TRAIN1 training set of 66%.

The PCC on the tests sets are comparable between the two networks. The network trained on the TRAIN1 test set scores a 70% and 51 % overall PCC on test sets TEST1 and TEST2, respectively. The network trained on the TRAIN2 test set scores a 73% and 53% overall PCC on test sets TEST1 and TEST2, respectively. From this, it is evident that training on the TEST1 test set does not allow the network to discover patterns inherent in the data any better than training on the TEST2 test set. Yet, there is an obvious difference between the networks ability to classify non-border pixels and non-distinguished pixels. Apparently, the network trained on the TRAIN2 training set is able to filter out the noise endemic to the border pixels, or at least that noise is overshadowed by the signal in the non-border pixels.



a. b. c. d.

Fig. 1. A pictorial comparison of the categorization performance of the 5 hidden node network for category types 9 and 14, dense hardwood forest and residential respectively. Network was trained on top half of image. Sub-figures *a* and *b* show the pixels with ground truth types 9 and 14. Sub-figures *c* and *d* show the pixels that the 5 hidden node neural network classified as types 9 and 14. Pixels positive for the specified type are shown black (small white dots on black are meaningless), all other pixels are shown clear. The irregular border of the ground truth is shown by a lighter black background (*a* and *b*). That border area is filled in by the network because spectral information is available for a larger area (*c* and *d*).

BEST COPY AVAILABLE

Table 4
Percent correct characterized (PCC) of the training sets and the test sets after training with the training sets TRAIN1 and TRAIN2. Each of the 5 hidden node networks is trained for 10,000 epochs with the momentum factor set to 0.5 and the damping factor set to 0.5.

	Network Trained on TRAIN1				Network Trained on TRAIN2			
Type	TRAIN1	TRAIN2	TEST1	TEST2	TRAIN1	TRAIN2	TEST1	TEST2
1	-	-	-	0	-	-	-	0
2	4	1	0	0	0	0	0	0
3	-	-	-	-	-	-	-	-
4	96	78	87	70	96	81	91	73
5	0	0	0	0	0	1	0	0
6	0	0	0	0	0	0	0	0
7	45	29	0	0	10	11	0	3
8	100	65	81	58	100	65	87	67
9	80	61	86	74	88	71	92	81
10	95	70	82	57	99	76	82	66
11	59	49	50	45	29	29	28	25
12	27	9	12	8	2	10	4	6
13	71	35	100	43	0	22	0	33
14	79	76	86	75	52	51	65	53
15	-	-	0	0	-	-	0	0
16	0	0	0	0	70	63	67	73
17	1	0	0	1	0	0	0	3
Overall	66	49	70	51	65	48	73	53

Preliminary Conclusions

From these results it can be concluded that information is available in the raw spectral values concerning the ground truth classes into which individual pixels can be categorized. The limited training set precludes any conclusions about how accurate a neural network could get with this data. With some qualifications, a comparison can be made between the results of this study and and results of the earlier study for which the ground truths were originally collected [WIL84]. Although that earlier study used six bands of Thematic Mapper imagery compared to only four for this study, that study used imagery taken on November 2, a time "far from optimal for general category discrimination" [WIL84]. The current study used only 4 bands because at the time the image was taken, in July, 1982, the other instruments were not yet functioning. Because the earlier study used pixels from 9 sub-regions, only the first sub-region being used in this study, that earlier study had more training pixels than this study: 1600 training pixels and 600 test pixels for each class were chosen (see Table 1 for the number of training pixels in this study). When that early study used an iterative, point migration clustering algorithm with no editing of the training statistics, the overall PCC was 36.7%. When the analyst was allowed to interact with the computer to edit training statistics, then classification accuracy rose to 62.0%. When the 17 categories were aggregated into five categories (water, crops, pasture and grass, forest, and urban) and no editing of the training statistics was used, then classification accuracy was 65.7%. Because the neural network does not require any user interaction during training it is appropriate to conclude that this approach, with an overall PCC of 52.1% for the 2 hidden node network after 10,000 epochs, is quite favorable, despite the fact that the LANDSAT images were taken at different times for the two studies.

Future Directions

Because of our eventual goal to use this system for real time data characterization and categorization, input data should undergo minimal preprocessing. Among other uses, preprocessing can change the data into forms amenable for neural network processing, smooth out extremes in spectral flux values, and provide derivative measures of the data such as texture. That first alternative, changing the form of the data by representation, is frequently done by transforming the data from the spatial domain to the frequency domain. Derivative measures could use image segmentation. In this regard, examinations of a segmented version of the image will be made. Tilton [TIL88] has developed a parallel region growing segmentation method which has been used to segment the image into 207 subregions. That process substituted the spectral data for each pixel with the average spectral data for all pixels in that pixel's sub-region. Thus, the average value of the spectral bands for pixels in the sub-regions will be used as input to the neural network. Other derivative measures of these regions, such as texture, can also be tested for their usefulness.

Another possible direction is to perform a fine discrimination by a sequence of course discriminations. If a network can consistently discriminate between groups of classes (superclasses) or individual classes, a hierarchy of such neural networks could be used to extract and refine discriminations between ground truths. For example, a network could categorize 5 mixed classes into 3 superclasses, one superclass consisting of one class, the other two superclasses consisting of 2 classes each. Successive networks could further refine each of the two superclasses into their respective class constituents. This is particularly suggested by the ability of the 1 hidden node network to successfully distinguish types 4 and 13 (see Table 2). The decision of when, and how, to apply different kinds of networks could eventually be implemented by an expert system.

In order to extend this paradigm to work on multiple images taken of different locations and from different instruments, the use of absolute spectral data must be modified. The network will have to use relative shape of the spectral flux curve. One possible way to implement this is by a form of lateral inhibition among the input layer nodes.

Bibliography

[AND76] J. R. Anderson, E. E. Hardy, J. T. Roach, and R. E. Witmer, "A land use and land cover classification system for use with remote sensor data," *Geological Survey Professional Paper 964,* United States Government Printing Office: Washington, 1976.

[CAM88] W. J. Campbell, L. Roelofs, and M. Goldberg, "Automated cataloging and characterization of space-derived data," *Telematics and Informatics,* Vol. 5 No. 3, pp 279-288, 1988.

[FOG88] R. J. Fogler, R. L. Williams, and L. D. Hostetler, "A billion connection per second feedforward pipeline for computer vision applications," *Abstracts of the First Annual INNS Meeting,* Boston, Sept. 6-10,1988.

[RUM85] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," In D. E. Rumelhart, J.L. McClelland and the PDP research group (editors), *Parallel distributed processing: Explorations in the microstructure of cognition. Volume I: Foundations.* Cambridge, MA: MIT Press, 1986.

[TIL88] J. C. Tilton, "Image segmentation by iterative parallel region growing with applications to data compression and image analysis," *Proceeding of Second Symposium on Frontiers of Massively Parallel Computation*, Fairfax VA, Oct. 10-12, 1988.

[TIL89] J. C. Tilton, personal communication, 1989.

[WIL84] D. L. Williams, J. R. Irons, B. L. Markham, R. F. Nelson, D. L. Toll, R. S. Latty, and M. L. Stauffer, "A statistical evaluation of the advantages of LANDSAT thematic mapper data in comparison to multispectral scanner data," *IEEE Trans. on Geoscience and Remote Sensing,* GE-22, pp. 294-302, 1984.

BLANK PAGE

A RAPID PROTOTYPING/ARTIFICIAL INTELLIGENCE APPROACH TO SPACE STATION-ERA INFORMATION MANAGEMENT AND ACCESS

Richard S. Carnahan, Jr.
Stephen M. Corey
John B. Snow

Martin Marietta Information & Communications Systems
Denver, Colorado

1. ABSTRACT

This effort has focused, and continues to focus, on applying rapid prototyping and Artificial Intelligence techniques to problems associated with Space Station-era information management systems. In particular, our work is centered on issues related to (1) intelligent man-machine interfaces applied to scientific data user support and (2) the requirement that intelligent information management systems (IIMS) be able to efficiently process metadata updates concerning types of data handled. The advanced IIMS represents functional capabilities driven almost entirely by the needs of potential users. The amount and complexity of scientific data projected to be generated by Space Station-era projects (e.g., Eos) is likely to be significantly greater than data currently processed and analyzed. Information about scientific data must be presented to potential users in a clear and concise way, with support features being incorporated to allow users at all levels of expertise efficient and cost-effective data access. Additionally, since data modifications and additions will frequently occur, mechanisms for allowing more efficient IIMS metadata update processes must be addressed. To this end, our work has examined the following aspects of IIMS design: (1) IIMS data and metadata modeling, including the automatic updating of IIMS-contained metadata, (2) IIMS user-system interface considerations, including significant problems associated with remote access, user profiles, and on-line tutorial capabilities, and (3) development of an IIMS query and browse facility, including the capability to deal with spatial information. A working prototype has been developed and is being enhanced. Future work will attempt to clarify a number of issues which have emerged from our present efforts, particularly concerning IIMS information base structure and its relationship to user-support and distributed, heterogeneous database access.

2. INTRODUCTION

This paper represents the culmination of our early examination of the areas of advanced information management and access using rapid prototyping and Artificial Intelligence (AI) programming techniques. In one sense, it is inaccurate to have only one section with the heading 'Results'. The entire paper reflects our current thinking concerning issues surrounding future IIMS operations and development, and all of what is written here is a direct result of our early prototyping efforts. To date, our work has served as a validation of the approach we have used since the initial prototype has helped to crystalize our thinking and drive out more specific system requirements. As a result of what we have accomplished, we are now in a position to move forward. Section 3 details our interpretation of global requirements advanced information systems will necessitate, and our understanding of current system limitations that compel the use of advanced concepts in both system design and implementation. Section 4 discusses our approach to implementing a prototype IIMS using the application domain of large science databases. Section 5 presents the current state of our prototype and concluding remarks concerning implications of our work.

3. BACKGROUND AND PROBLEM

3.1. Current State of Space System Data Management

For decades, NASA and other agencies (e.g., NOAA, NCAR, EROS) have collected vast amounts of scientific data in support of Earth and space research. For the value of this data to be fully realized, it must be easily accessed in a timely manner not only by those researchers directly associated with a particular mission or project, but also by researchers who are involved in the domain of interest as well as those in related scientific fields. To complicate the situation, the amount of data being collected or generated is increasing exponentially and should continue to grow in the future [15]. The end result is that many researchers cannot use data they need because they do not *know how* to access the data, they do not *have* access to the data, or they do not *know where or if* the data exists.

Currently, data (we use the term 'data' to refer to all collected and generated data as well as information about data [*metadata*]) reside with the agency sponsoring the data collection effort or, in some cases, in large repositories. Some of these facilities publish catalogs from which the user can order data, but they usually provide no means for remote access. Such a policy has many obvious drawbacks; the most serious is difficulty in locating and obtaining all suitable data in a short period of time. If users are fortunate enough to find needed data, they must often convert the data into a form their algorithms can process before it can be studied. Even facilities or projects having computerized data access have many limitations when it comes to locating all relevant information, and usually lack sophisticated querying and user-system interface techniques. These systems tend to be built in such a manner that makes access very difficult by someone unfamiliar with the data. With the projected increase in amounts and complexity of acquired data, the need to have data available to all researchers, not just those directly involved, and the increasing need to understand relationships among data, it is rapidly becoming impossible to access the most suitable information.

3.2. New Directions

NASA is progressing with new information gathering programs that will produce significant amounts of new information in the 1990s (e.g., EosDIS, Hubble Space Telescope, CDOS [11]). NASA has realized that, with recent advances in computerized information technology, there are opportunities to enhance the useability of collected data. There is an understanding that data and information management systems required to support the Earth and space research community will need to be complete research information management systems [10]. One fundamental requirement of these new systems is to allow flexible and transparent access to geographically dispersed, heterogeneous databases. Access methods will need to use advanced information management techniques including appropriate search and AI methods. The goal of such systems is to elevate the scientist from having to concentrate on the details of accessing and preparing data for analysis to concentrating on his research [10]. In response to these problems, NASA has begun the Intelligent Data Management Project to begin research into and implementation of advanced information management systems [14]. In addition to this NASA effort, many other researchers have begun to address issues surrounding the IIMS.

3.3. Intelligent Information Management Systems

The goal of future Earth and space information management systems is to intelligently manage data in four areas: (1) data collection, (2) data processing, (3) database management, and (4) accessing and archiving [9]. In light of the total information system perspective, research reported in this paper has concentrated on the development of intelligent user-system interfaces and intelligent information integration and access. Following sections will focus on these areas. However, before discussing features of the IIMS, it is important to understand some of the more salient limitations of current information or data systems that provide for computer access. Typically, such systems rely on traditional database technology that has several disadvantages when providing for a complete research system. Some of the more important limitations are discussed in the following sections.

3.3.1. Query Limitations

The method generally used to extract data from current systems is querying the database through the use of a query language. To query the database, the user must *instruct* the system concerning the scope of data in which the user has an interest. After forming a sentence recognized by the particular query language in use (which, in general, is not natural language-like), the database system will return the requested data; the user may or may not be able to interactively view the data. In either case, the user must then determine, after some amount of processing, if the data satisfies his needs. Such a process may be repeated several times before data satisfying the user's requirements are obtained. The process of forming a query statement and then analyzing the data returned from the execution of this statement has several limitations.

Learning the Query Language. Before obtaining data from the database, the user must learn how to interact with the system; such learning is accomplished through the use of a query language (i.e., normally mathematically rigorous grammars that require the user, when requesting data, to express boundaries in a rigid syntax). These languages are typically difficult to learn, and their effective use often requires a high level of expertise. The user can easily become confused about how to express requirements for data sought, and frequently, needed data cannot be located, or data that is not needed is received because of difficulties inherent in the forced, unnatural expression of the request.

Knowledge of Data Structure and Relationships. After the user has learned the query language to a degree which permits forming queries, the next step in the process is comprehending the structure of the database. This is made necessary because the query language requires that the user inform it where the data resides within the database structure. In a relational system, the user must know the names of relations and attributes by which data is stored. A second limitation is that data in these systems are usually organized for speed of access and without consideration either for relationships among the data or concepts inherent in the domain of the research. Therefore, to access data relating to a basic concept within any research domain might require a single query spanning several relations and performing many calculations. When using all but small databases, such a requirement places a large burden on the user, and its execution is usually beyond the capabilities of researchers dealing with large applications. For example, the Crustal Dynamics Data Information System database consists of 144 relations and 679 attributes [14]. To use a normal query language, the user would have to know all relation and attribute names, as well as relationships between attributes, and be able to translate domain concepts into the query language. Obviously, to do so for the Crustal Dynamics Data Information System database would be extremely difficult.

Query Reformulation. In some cases, queries will not extract the expected data, requiring the user to resubmit a modified query which, hopefully, will return better results. Typically, no aid is given to the user in comprehending why the original query did not return the required data, and thus, it may not be clear how to appropriately modify the query.

3.3.2. Interface Limitations

In addition to limitations discussed above, current database systems often fall short of providing needed capabilities in the area of user-system interface. Systems today are often text-oriented with little facility for graphic representation or direct manipulation of screen objects. User-system interface requirements deemed necessary for an IIMS are detailed in section 3.4.2.

Beyond limitations already discussed, there are other querying process and user-system interface technology limitations of current systems. However, the discussion provided above has shown the general unsuitability of a traditional approach for the large, complex databases like those that are and will be encountered in Earth and space research. Rather, new capabilities and new approaches are required.

3.4. Functional IIMS Requirements

Several other information management concepts and technologies have emerged over the past several decades that can be brought to bear on current system limitations. However, before we can evaluate the appropriateness of any of these approaches, a formal definition of IIMS capabilities must be presented. Our effort has concentrated on how to provide the user with the capability to locate and manipulate desired information while at the same time allowing an efficient and flexible mechanism for providing automatic updates to the information data/knowledge base. The IIMS functional description presented in following sections reflects this conviction. Requirements for an IIMS that our work has addressed can be divided into two categories: (1) foundations of the system; that is, those parts of the system not directly accessed nor seen by the user, and (2) the intelligent user-system interface, through which all interactions between the system and the user are managed.

3.4.1. IIMS Foundations

The foundations of any system which accesses data are its underlying information structures and its methods for manipulating those structures; this is particularly true for the types of large systems we are addressing. In these systems, data will likely be stored in heterogeneous computing environments. The access system will have to be able to handle multi-users and allow timely updating of the various databases and knowledge structures without interrupting users of the system. Because of the significantly large amounts of data entering the IIMS on a daily basis, updates to data and knowledge bases will need to be automatically performed. Any attempt to handle such updates using manual techniques would soon overwhelm available personnel. Some projected systems will require that data be incorporated into the system and made available to users within a few hours. Meeting these requirements depends on the design and implementation of information structures and intelligent techniques allowing for automatic update.

Access to heterogeneous databases and information structures must be transparent to the user. To provide transparency, the query subsystem must be able to decompose a query the user has formed into a set of queries that access appropriate databases in their associated query languages. This subsystem must be able to manage multiple sources of similar data types and, depending on the cost of accessing each alternative and the loads on and limits of respective systems, access the most appropriate database. Additionally, queries might involve data processing by special algorithms. The query subsystem must understand where copies of such algorithms reside and processing limitations of the various host systems. If part of the data is not available, or the cost of query execution is above a certain threshold, the user should be notified. If a query can be executed, results returned must then be combined and processed to provide the answer to the original query, and then converted into the correct presentation form. Future information systems are currently planned as long-lived systems, and it is generally accepted that they should be able to adapt to short- and long-term variations in operating conditions and operate with low maintenance costs. To satisfy these conditions, data and knowledge structures must be able to be modified to interactively incorporate new types of data without interruption of normal system operation. Any inflexibility in this respect could render the system unusable when operating conditions change.

3.4.2. Intelligent User-system Interface

The role of the intelligent user-system interface is critical to the success of the entire IIMS. Even if the foundations are successfully laid, it does little good if the user is not able to easily and efficiently access information. Within the normal pursuit of research goals, the user-system interface should stimulate the user to explore and use the whole system with all its capabilities, and should always be an aid, not a hindrance, to the user. While these goals have proved difficult to satisfy in a general sense, much has been learned from research in their pursuit. Some features discussed in following sections have been shown to be useful in achieving these goals, while others are just now beginning to be explored.

User Profiles. One technique that can contribute to providing a supportive interface is using knowledge about the user to make system access more efficient. Most generally, each user has different areas of interest and levels of expertise (both within the user's domains of interest and knowledge of the IIMS) that can change over time. To provide the best environment for different users, the system must be able to adapt to such variable conditions, and also understand information about individual users or groups of users. We have found it useful to group such information into two categories: context sensitive expertise and goal comprehension.

To provide ease of use for experienced as well as novice users, the system must understand the level of domain and system expertise at which the user is operating. A system may be easy for a novice to use but may be very difficult, for various reasons, to use for an expert. In particular, as a user becomes familiar with certain parts of the system, support tools necessary for a novice user are no longer required. For example, a menu-based system may be adequate for a novice since it provides a type hierarchy for locating available operations. However, menus can be time consuming for an expert unless shortcuts, such as commands bound to keys or macros, are available. Despite allowing for some limited user growth in system operations, menus alone are not sufficient for the type of system proposed here since they do not recognize the expertise of the user. To be truly effective and useful, the system must understand the expertise of the user within the context in which he/she is currently operating; understanding which is required since users' expertise varies for different parts of the system. To address user expertise by measures such as time-of-use is simply too course grained. The growing subsystem operating expertise of the user must be reflected by modification of offered support features.

Future information management systems will contain many different types of data, ranging from accounting and collected scientific data to information about instruments and users. As a consequence, the system must be able to accommodate operating variations resulting from users with different interests, goals and expectations about using the system. Two particular concepts are useful in understanding the need to manage such variation: intelligent views and group-type hierarchies.

Intelligent Views. For users to be most comfortable with data access and presentation, the IIMS must be able to communicate with the user using familiar terminology, concepts, and data organization. That is, the system must relate to the user in a manner consistent with the user's domains of interest. Intelligent views are one technique used to incorporate knowledge about the user's domain and concepts of data organization. For example, a user who is interested in cost figures for certain organizations, or performance of the system, is not necessarily interested in all data the system contains; presenting all data would only serve to confuse and inhibit the user. Hence, only data or information within the user's current domain of interest should be displayed. In addition, information should be presented to the user in a manner consistent with the type of data being displayed (e.g., accounting information being displayed in the form of graphs or charts). Users may select standard views defined for various application areas, and the system should allow existing views to be modified or new views to be created.

Group-type Hierarchies. Since interests and goals tend to be related within groups of users, the IIMS should be able to handle a user group-type hierarchy and associated inheritance capabilities. This would allow for users to inherit certain characteristics of a particular group (e.g., science users, system operators) while allowing for personalized modifications.

Help Facilities. Another feature required to provide a complete IIMS is an extensive *help* facility. *Help* should be available no matter what the user is attempting to accomplish, and all *help* should be context sensitive. For example, a general explanation of the current system should be provided when no specific subject is selected, and when a specific subject is selected, detailed information should be provided. *Help* facilities must be responsive to levels of user expertise in the selected subject and should allow for exploration of related information. A general overview of information concerning system operation and data available should be provided to novice users (tutorial information), while expert users are immediately allowed to select more detailed information. The novice user can progress to more advanced operations by navigating through the information base associated with any particular subject area. There should be no difference in presentation style and methods used to provide *help* about system operations or *help* about data. For example, if a user requests more information about a particular instrument, the user should be able to navigate to a drawing of the instrument, descriptions of its specifications, other related instruments, and possibly information about the manufacturer. Information required to answer users' requests for *help* is structured no differently than any other information stored in the information base. Finally, *help* facilities include an intelligent tutorial system (see section 4.3.3) which can fully explain the IIMS and its capabilities. This tutorial system is linked to the rest of the *help* system and uses much of the same information as other *help* facilities, allowing the user to obtain a tutorial on a current operation without specifically entering the tutorial subsystem.

Intelligent Information Manipulation. The IIMS must provide an interface to the system's query capabilities, and should allow the user to transparently form queries without having to learn or use a mathematical query language. This type of query formulation system insulates the user from having to know (1) the structure of the data and (2) what information is contained in the database. As the user forms the query, the system should indicate what data is available at each stage of the query formulation process. The user should be allowed to form queries using conceptual information processing techniques provided by intelligent views. Using this methodology, the user should be able to form a query by inserting range requirements, spatial, and temporal indications. Moreover, the IIMS should be able to store and provide for query modification and reuse, as well as allow for browsing the information base. Ideally, browsing and query formulation capabilities should be combined. Finally, the system should be capable of displaying different types of information in graphic formats.

3.4.3. Flexibility and Extensibility

As stated earlier, data and knowledge structures must be able to be modified to handle new data types without interruption of normal system operations. Furthermore, the user-system interface must be flexible and extensible. Different users have different preferences for the organization of the interface and use of available system support tools, and need to be provided with a measure of control over interface presentation. Finally, the capability to access algorithms and other external programs (e.g., Geographic Information Systems) must be allowed without the need for reprogramming. In general, flexibility and extensibility of the system must be carefully considered in all design and implementation decisions.

3.5. Other Information Management Techniques

Research in information management has resulted in many useful systems and techniques. Each of these systems and techniques has focused on certain issues or types of data but none have addressed the entire range of IIMS functional requirements. Database research has provided several techniques and systems, including (1) the ability to efficiently provide multi-user access to distributed databases, (2) indexing techniques for various types of data, including spatial and temporal, (3) transaction processing and recovery techniques to ensure data integrity, and (4) dynamic data management. Hypermedia systems provide information and techniques on (1) navigation through a large information base, (2) integration of various types of media, and (3) representation of relationships among data. Finally, AI systems have provided valuable techniques for (1) representation of relationships among data as well as other knowledge representation capabilities, (2) data driven processes, (3) planning and scheduling systems and (4) intelligent user-system interface capabilities.

Taken alone, none of these approaches can satisfy all requirements for the next generation IIMS. In general, proponents realize the limitations of their approach and are beginning to work on removing these restrictions. Database researchers, for example, are examining ways to incorporate data relationship information while continuing to research new indexing techniques for different types of data. The field of hypermedia is beginning to examine incorporation of search techniques to allow for more efficient navigation within the information base. Expert database systems are studying the use of indexing techniques to improve system performance. All these approaches are addressing the problem of accessing information in an efficient manner, and although they began by looking at different types of data, research directions are resulting in movement toward a common viewpoint. We believe that the next generation IIMS should combine experience gained in each of these fields of study; not an easy task since the approaches are not always compatible and some projected capabilities of the IIMS are still beyond current technology. However, the use of appropriate techniques from each of these approaches, combined with lessons learned from user-system interface research, should provide a good starting point for implementing such a system. As discussed above, we have taken an eclectic view of the IIMS and the following section details our current approach to developing the IIMS prototype, and our present thinking concerning several issues surrounding future IIMS development in relation to the chosen application of large scientific databases.

4. APPROACH

4.1. IIMS Operations Concept Development

Advanced information system operations environments will be required to perform and provide support for a multitude of functional capabilities, ranging from data capture to teleoperations. Future information system design will be largely influenced by the demand for extensive and sophisticated data acquisition, handling, processing, management, and access facilities. It is critical that data not only be obtained in a timely and cost-effective manner, but that acquired data be made readily available to potential users. While issues associated with efficient data acquisition are significant, it is the latter requirement that necessitates looking beyond traditional and currently available technologies. Issues of user transparency, in terms of command and control, planning and scheduling, processing, and information management and access demand application of new and innovative computational programming paradigms.

Several potential factors influence advanced information system operations. For some functions (e.g., command and control, planning and scheduling), the information system may provide a direct link between the user and the flight system. It is important to realize that cost-effective future information system operation will require that many current, man-intensive operations functions will either become part of the flight system functional repertoire or will be handled autonomously on the ground. Again, command and control, planning and scheduling, system management, and fault detection/isolation are excellent examples of potential automation candidates. Future information system features of telescience, user transparency, and remote distributed access will require levels of system operation sophistication not currently available. The purpose of our work, then, has been to closely examine several projected, advanced information system capabilities in light of the potential application of advanced programming technology.

As suggested earlier, it seems apparent that two major issues confront the design and implementation of the IIMS: first, how to design an effective, transparent user-interface which not only allows for multiple remote user access using different hardware/software but also provides intelligent support features for helping users with different levels of expertise explore, identify, and obtain appropriate system data, and second, how to design an efficient and flexible mechanism for allowing automated incorporation of (1) new data types, (2) information about the new data, and (3) appropriate links with other metainformation. The prototype IIMS will provide a proof-of-concept demonstration of the applicability of AI programming techniques and tools to the domain of distributed data access and management. The prototype IIMS will be able to (1) accept multiple user data requests from different types of remote user-interfaces, (2) intelligently guide the user to an understanding of the types of information and data available, and (3) accept and integrate information concerning data updates from multiple, heterogeneous databases.

4.2. Application of AI and Rapid Prototyping to the IIMS

Before discussing our approach to determine the potential applicability of AI technology or advanced programming techniques to advanced information systems, it is important to understand some basic assumptions we are making concerning AI:

Assumption 1. The term 'AI' is not particularly useful for describing the many potential advanced programming technologies which may be applicable to advanced information systems. It would probably be more appropriate to determine the potential applicability of AI in terms of more specific sub-disciplines such as expert systems, intelligent resource allocation, intelligent user-system interfaces, intelligent networking, intelligent training, intelligent information management/access, simulation, automated programming, natural language, and machine learning.

Assumption 2. It is important to distinguish between (1) application of AI technologies to advanced information systems and (2) use of advanced programming techniques (e.g., object-oriented programming, production systems) associated with AI to facilitate development of more traditional types of applications. For example, work of ours has shown an almost eight-fold decrease in code development time and an almost four-fold cost savings when using object-oriented programming and production systems to develop simulations of Spacelab payload experiments for purposes of payload crew training [3].

Assumption 3. Although initial decisions concerning potential advanced information system candidates for AI technology application can be made using specific, paper/pencil criteria, rapid prototyping will likely provide a much clearer answer concerning the potential adequacy of certain technologies for information system applications. This is particularly true when attempting to apply advanced programming techniques and environments to more traditional types of programming applications.

Assumption 4. Although we believe appropriate application of AI technologies/advanced programming techniques would provide significant benefits to certain areas of advanced information system operations, we also believe that such technologies should be applied *only* when appropriate. Often, a hybrid of traditional and AI programming approaches is useful for optimal system performance as well as system modification and enhancement.

Our basic approach to determining the potential applicability of specific AI technologies and advanced programming techniques to advanced ground system functionality is highly iterative, with domain experts involved at each step of the process. Using surveys of existing systems, initial criteria, and expert advice, potential application candidates are culled from the total domain and after further examination, a select number of applications are chosen for proof-of-concept demonstrations. Rapid prototyping these applications using AI technologies and advanced programming techniques is, again, an iterative process, serving both to identify candidates for system implementation as well as to help drive out more detailed system requirements. An important feature of this process is that initial candidates can be quickly chosen, and application of rapid prototyping techniques provide the mechanism for rapidly narrowing the potential field of candidates for actual system implementation. Of course, the type of prototype development environment used is crucial to the successful application of these techniques (section 4.5 describes the hardware and software configuration we are currently using for our effort). Given results from the prototyping effort and input from domain experts, a final set of applications is chosen for actual system development.

4.3. IIMS Prototype Implementation

To date, our work has suggested several potential IIMS functional requirements that provide good AI technology application candidates. Figure 1 illustrates our current view of basic IIMS functionality and indicates those areas we believe will most likely benefit from the appropriate application of AI technologies. Although our prototyping work has focused on a limited set of these, we believe that all are appropriate areas where advanced techniques could be applied. Discussion of our present thinking concerning implementation of each function is given below.

4.3.1. User-system Interaction

Rather than a set of separate modules which are activated when required, we view all interaction between the user and the IIMS to be handled by a single, integrated system. All user-system operations are physically and conceptually linked to the system as a whole, and as a result, *help* facilities are handled in the same manner as browsing the data or knowledge bases. Such a technique maintains a consistent user-system interface and simplicity of design, while making it easier for the user to learn and use the system. When all data is treated the same, accessed in the same manner, and traversal from one set of data to another is transparent, the user does not have to learn different methodologies to interact efficiently. In addition, context switching is not dramatically experienced, thus reducing confusion normally associated with such transitions.

To provide rapid access to an environment whose structure can be transparently and dynamically altered, we employ a conceptual structure of *metadata*, or information about the data, as the main information structure. All information including non-scientific data (e.g., *help* information, tutorial information) and information about the scientific data contained in the system is stored using the same conceptual structure. Actual science data, stored in distributed databases, are accessed only when a query is being resolved or specific information about the data is required. *Metadata* structures are automatically generated and maintained by a set of expert systems which incorporate conceptual information about different domains and relationships among data. All information base (real data and other knowledge) update messages are processed through these expert systems which then make appropriate modifications to the knowledge structure. We choose to use this approach so that navigation of the information structure will not be slowed by the execution of a large expert system; we can still use the powerful techniques expert system technology provides for automated knowledge extraction and information updates.

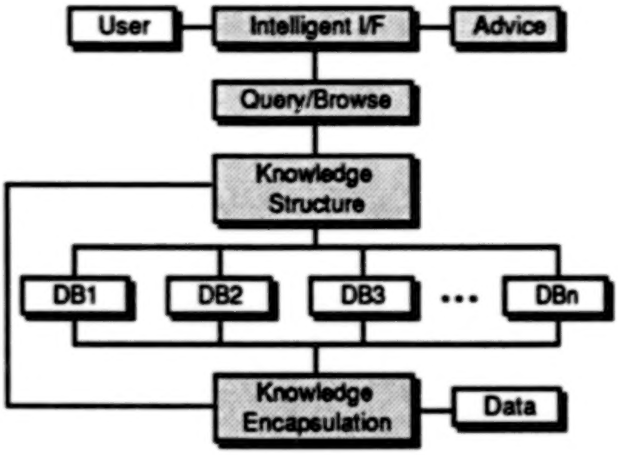


Figure 1. IIMS Functional Architecture

For users to be most comfortable with the task of locating information, data presentation must occur in a familiar manner. Appropriate *metadata* structures should accurately reflect domain concepts and use domain terminology. As viewed by the user, such structures cause domain specific variation in the knowledge structure, although no change occurs in basic relationships among data which form the global structure. Providing this capability, relationship information is provided by physical, typed links with concept dependant, relevance weightings, allowing for representation of the notion that certain links have a high degree of relevance for certain domains and have no relevance for others. As the user traverses the knowledge structure, links available for selection by the user are chosen based on (1) the relevance weighting factor within the context of the current domain and (2) the path the user took to arrive at the current node, and (3) a user defined threshold value. Since weighting factors allow for a range of values between 0 and 1, a wide range of concepts can be represented, allowing for the inclusion of fuzzy concepts and searches. We will use relevance weights to provide for inclusion of relevance factors to represent different levels of user expertise.

We provide mechanisms, which we call intelligent views, that allow the user to define and modify knowledge structures (In the future, they will also allow for the specification of plans and other intelligent support features). The user can then select which view is considered most appropriate for the current task and activate it. This modified view is then layered on top of the global knowledge structure (this does not alter the global information base; only the user's view of it), changing the user's view of the global knowledge structure and providing the capability to explore the data in a familiar manner. Intelligent views can be shared by users, allowing the facility for domain experts to define a series of views to be used by others. The method for creating these views is the same as that used in navigating the knowledge structure, with a few extensions to allow for (1) creating new or combining already existing nodes, (2) including plans or other knowledge based programs, and (3) altering weighting factors.

Query formation is achieved by indicating items of interest through the user's navigation within the knowledge structure. As stated above, choices offered to the user at each location are determined by calculated relevance factors. At certain locations within the knowledge structure, users can indicate Boolean and range specifications for certain attributes, with potential extensions to allow for fuzzy characterizations. The manner in which range specification is indicated depends on the type of attribute. For example, a map of the world is displayed to allow the user to graphically specify the spatial area of interest. The user is also allowed to select predefined regions (system or user defined) or, if desired, enter map coordinates. Facilities for scrolling and zooming provide the user with desired levels of detail. With his methodology, there are no perceived differences to the user between *help* and scientific information. Both are information for which links have been defined. Therefore, context sensitive *help* is provided in most locations by links to appropriate information.

Using navigation techniques to allow formation of queries eliminates many of the traditional problems associated with standard query formulation: first, the user always knows what data is contained in the database and what next selections are possible. Second, since this process is constrained to knowledge the system contains, only valid queries can be formed. Finally, the user does not need to learn a query system but can "converse" with the system in the terminology of the chosen domain. As a result, this method provides facilities to query the database and browse through the database, locating information of interest or exploring related topics. While providing several advantages, there are also problems with this type of query formulation. Four we are currently concerned with are (1) the problem of getting "lost in space", a problem encountered by hypermedia systems when navigating a large body of information, (2) elimination of extraneous operations encountered by users familiar with the data, (3) providing non-navigation movement techniques, and (4) smooth integration of Boolean and range qualification specifications.

Lost in Space. When users are allowed to navigate a large body of information which provides a rich environment for exploring related topics, disorientation, due to inadequate cues concerning current location, is typical. Standard graphical techniques that display the information base as a group of nodes and links can become unusable because of the density and complexity of displayed links [4]. We are using several techniques to address this problem. Intelligent views help reduce the information space and the amount of information directly accessible at any one time, since not all links proceeding from any particular node are relevant to the current domain context. This technique aids in reducing complexity, but, in many cases, will not reduce it enough to allow for total reliance on graphical display techniques. One method we are exploring is to provide graphical structure overviews to show detail based on both distance and relevance. Consequently, the farther away the user is from a node, the higher the relevance threshold is for links, thus reducing the number of links displayed. Furthermore, there is an inherent, hierarchical organization in much of the information space that can be utilized, aiding the selection of nodes to be eliminated from the display. As a result, the number of a node's "children" (this term is used loosely) not displayed depends on their distances from the current node, appropriately reducing the amount of information displayed while providing enough location aids to be of use.

Elimination of Extraneous Operations. Users tend to frequently state the same or closely related queries. This inclination could become cumbersome if the user was forced to traverse the information base in the same manner each time; the user would be forced to frequently enter repetitive operations and would probably become disillusioned with the system. We are addressing this issue using several alternative methods. The user will be allowed to save partial or complete queries that can be displayed in graphical form, modified, and then resubmitted. These saved queries can be treated as an intelligent view by highlighting query choices on the backdrop of the active view at the time of query creation. Such a procedure allows the user to modify specific query values or include or delete other items.

Alternative Navigation Techniques. While navigating through the information base at the same time as forming a query or browsing, the user may wish to move to another location that cannot be directly accessed from the current location. The absence of this capability would force the user to find some path through the information base, a time consuming and often difficult process. Initially, we will allow two methods to provide this facility: (1) via the mouse, allow selection of a node from the graphical knowledge structure display, and (2) allow the user to enter a keyword or phrase as a search criteria, resulting in a display of all nodes satisfying this criteria. The user will then be able to choose any appropriate node.

User Profiles. We have already discussed some potential types of user or group information that can be defined and manipulated (e.g., intelligent views and saved queries). This information along with other information about the user (e.g., group type, domains of interest, expertise when dealing with different parts of the system, interface preferences, accounting information, access privileges) will be kept in a user profile. Defining the user for the IIMS so that the system may be appropriately configured will aid in providing easy and useful user-system interaction. In general, users will be considered as objects and user hierarchies will be established, allowing for inheritance of abstract user information while providing for individual user preferences.

Results Display. The final part of intelligent interaction with the user is the display and manipulation of results. At this point, it is unclear what types of displays and interactions would be most useful. It is understood that graphical displays, including image displays, will be required, but the extent of potential results display functionality has yet to be determined. It is fairly obvious that much depends on the type of data to be displayed (we have already alluded to the notion of display representation being closely tied to the type of data being displayed). We plan on interviewing potential users of these types of systems to determine what types of display are required.

4.3.2. Knowledge/data Encapsulation

All data stored in IIMS databases are input from various sources, and when new data becomes available, the IIMS catalogs and directories must include information about the new entry. This process is accomplished through the mechanism of an update message which informs the IIMS that new data exists, where it resides, and a number of appropriate attributes which describe the data. Additionally, information concerning associations and relationships between the new data and existing data must be generated. In general, the knowledge structure describing data that exists must be modified to accept the new information. Clearly, given expected high data rates and volumes, it would, at best, be inefficient to process such metainformation in a manual fashion. Hence, the requirement for an automated *metadata* update system is essential.

Upon receipt of the message indicating new data has been input, new *metadata* is processed and inserted into the IIMS *metadata*base structure. As suggested above, such processing involves understanding the new data so that appropriate relationships between new *metadata* and other *metadata* already contained in the *metadata*base can be established. The speed at which this process can be accomplished depends on (1) the type of data received and (2) its potential relationships to other *metadata*. Establishing these links can be technically challenging, and the success of any initial capability and the subsequent learning of future relationships from patterns of use during user-interaction sessions is key to efficient *metadata* access. The most promising approaches to handling metainformation inference seem to be closely allied to object-oriented database technology [7]. We expect that one or more expert systems will be required to provide the inferencing capability required for such a system. Initial estimates of potential data input frequency and associated update messages will need to be made since it could be possible that infusion of new data to the IIMS will outpace its ability to accomplish necessary processing of the *metadata* in real-time. The IIMS will therefore require the ability to buffer update messages until such time that processing can take place. An alternative solution is to immediately update the IIMS's databases and perform required *metadata* processing on a delayed basis.

Moreover, *metadata* extraction algorithms will have to be provided to the IIMS for each of the types of data the IIMS must know about. Since most of these algorithms will be application specific and subject to modification, a method must be established to handle this dynamic environment. New or modified algorithms will need to be incorporated in an efficient manner. A system will be required which allows specific format specification of the *metadata* to be received by the IIMS, data attributes which are important, attributes to pass on, any processing which must be applied to the attributes to produce additional attribute information (e.g., algorithms will be required to extract relationships among attributes). In some cases, *metadata* updates will need to be manually submitted when *metadata* cannot be automatically generated (e.g., the user may wish to suggest some alternative associations or relationships that are not obvious to the system). To make this process as efficient and consistent as possible, an intelligent forms entry system will likely be required.

The representation sophistication of data relationships directly influences the type of implementation necessary for IIMS data- and *metadata*bases. In the simple case, where only a few links are provided, a commercial database, along with a simple conceptual net or other knowledge representation technique, would probably suffice. At the other extreme, all data could be stored in a knowledge based system. The problem with the second alternative is that such a system may not be able to efficiently handle the volume of data projected for the future IIMS. The most probable implementation will be a commercial database to efficiently handle volume and query processing in addition to a knowledge representation scheme/expert system which helps determine relationships among the data. These two systems would work closely together to provide desired capabilities. It is not clear at this point just how much responsibility would be assigned to each system.

4.3.3. Advising/tutorial Capability[1]

Since the primary consideration used to determine the usefulness of any information management system is its ability to make information accessible to the widest variety of personnel, it is extremely important that the system be easy to use and provides the necessary user support. Several potential user support requirements have been postulated [2], including:

1. Facilitating understanding by specifying the contents and meanings of a collection of data as well as the relation between objects within the data;

2. Supporting approximate reasoning to infer conclusions that are not explicitly stated by the user;

3. Handling information demands for which the database is used routinely (macros);

4. Communicating with the user in plain English text;

5. Providing a physical and logical link between information contained in the meta-knowledge database and the actual data.

As indicated in section 3.4.2, we might add to these the notion of intelligent *help* or tutorial capability, allowing naive and expert users ease of system use without long learning times. Indeed, the intelligent *help* facility could either be on- or off-line, depending on the mode of operation in effect at the time. User profile capability would be very useful here, being used to expedite transactions and containing at a minimum the following user information: skill level in using the system, user areas of interest, conceptual views the user employs, type of terminal and software the user is operating, location of the user, communication lines which are available to send data, and accounting information. As with all *metadata*, the IIMS should be able to continually update this information as the user learns and interacts with the system, allowing interaction methods to be adapted as user needs or skill level changes. Our IIMS prototype implementation ties the user profile capability to the intelligent *help*/tutorial facility to allow more effective system use for each individual user. As outlined in section 4.4, using objects as a user-model representation technique provides exceptional flexibility in facilitating overall IIMS operations.

---

[1] A detailed discussion of Intelligent Tutor Systems is beyond the scope of this paper. Excellent overviews of the field can be found in [13,17].

In terms of actual advising, it is important to explicitly deal with the user's conceptual model of the IIMS [5,12]. Obviously, each user may have a different conceptual system model, resulting in confusion if the underlying system is not equipped to handle the varied expectations. Such conceptual model information could be associated with the user profile and used in several ways (e.g., design of user views, complexity and types of data/knowledge display techniques, query capability). Direct manipulation interfaces often provide the most effective means for learning a new system and helping expert users with ordinary day-to-day tasks (e.g., allowing the user to directly construct command macros used to speed system operations). The current IIMS prototype implementation relies heavily upon direct manipulation of system features to provide the user with necessary information and *help* concerning operations.

4.4. Object-Oriented Programming and the IIMS

The notion of representing dynamic systems as collections of interacting components with associated behaviors is intuitive, and provides a basis for designing more flexible and easily modified software systems [3,8,16]. Message passing among objects emulates potential system interactions since single messages can initiate any number of complex behaviors or other messages. Clearly, use of objects need not be limited to simple system component representation. They may also be used to represent other, more abstract processes and procedures (e.g., control processes, commands, *metadata* templates). Object-oriented programming paradigms are rapidly becoming a primary software development methodology used in a wide variety of applications. In particular, recent work in database and database management system (DBMS) technology has begun to seriously consider object-oriented data structures as a viable alternative to more traditional data representation techniques. Three primary advantages of object-oriented data models, when compared with more traditional models, have been postulated [6]:

1. Databases can be viewed as a group of objects rather than a set of relational tables;

2. Object-oriented data models support at least two types of data relationships: interconnections among data and generalization or subtyping of data types;

3. Integrity constraints are more easily implemented using object-oriented paradigms.

Additionally, there is an increasing trend to develop data models consisting of sophisticated, hierarchical data constructs that often share attributes or functional characteristics, resulting in highly portable, semantically-oriented database systems with several possible categories of relationships among objects (e.g., generalization [is-a], aggregation [a-part-of], and association as well as relationship qualification [1,18]). As suggested earlier, using objects as both a data and knowledge representation technique provides a data and information management system structure allowing for semantic interpretation of the data residing within the database. In addition to using objects as a representation technique for databases, several other IIMS functional requirements lend themselves to object application.

Displaying information is crucial to the ability of the user to understand not only what is in the database but also the relationships among the various types of data. Recent developments in object-oriented display systems (e.g., windows, graphic objects) are well suited to projected display requirements of advanced information management systems. As described in section 5.0, we have implemented an interactive graphics object system to facilitate both the display and representation of complex data and *metadata* structures. The system is used extensively in the prototype's query implementation and will be used to develop both online and tutorial capabilities as well as in the area of data presentation.

Implementing a system that can effectively respond to any particular user's needs is a considerable challenge. We have already stated the requirement for the IIMS to provide individual users with personalized environments in which to work. Object-oriented programming provides an appropriate structure for supporting such individualization. Since we anticipate that future IIMS users will fall into natural user hierarchies, it is useful to represent those users as an object hierarchy. Of course, almost unlimited user information may be associated with appropriate user objects, allowing for efficient and more effective user-system interaction. In particular, user views can easily be designed to reflect individual needs without fear of adversely affecting any other users.

4.5. The Mission Operations Laboratory

The current Mission Operations Laboratory (MOL) configuration is shown in Figure 2. As indicated, current programming is entirely microprocessor-based. The choice of a microprocessor-based computational and programming environment for the MOL was predicated on two major considerations: (1) the growing trend toward microprocessor workstation technology and (2) the recent availability of advanced, highly sophisticated dual and multi-processor micro systems. Our intent in designing the current MOL configuration is to provide an environment similar to those we expect to find in the next generation of information management systems, many components of which will be based on heterogeneous computing environments. The advent of true telescience will bring with it the capability to allow remote users, using many different types of hardware and software, transparent operations and data access. To provide the extensive user support required for efficient operations, high-level languages (e.g., Ada, LISP, Prolog) will need to be supported. Advanced, non-traditional programming paradigms will likely become commonplace since they provide the flexibility required to handle significant software modifications without significant impact or perturbation to system operability. In addition, application of AI technologies to several system functional capabilities necessitates powerful software development environments.

In the near future, the capability of the MOL to test various advanced concepts in the domain of information management will be significantly enhanced with the addition of hardware using both parallel processing and advanced optical disk technologies. First, the ability to apply different parallel architectures to features of an information management system will lead to a much better understanding of how such topologies might influence areas of user-system interaction as well as data handling, storage, and access. As indicated in Figure 2, our choice for a parallel development environment is a Sun 3/260 including multi-processor capability using transputer technology. The resulting message passing architecture is ideal for proof-of-concept design and test of several parallel topologies applied to any number of information management system functional domains.[2] Second, optical disk technology provides the capability required to test advanced knowledge and data storage concepts. Projected information system high data input rates and volume storage require that the impact of optical disk technology on both functional and performance requirements be carefully examined.

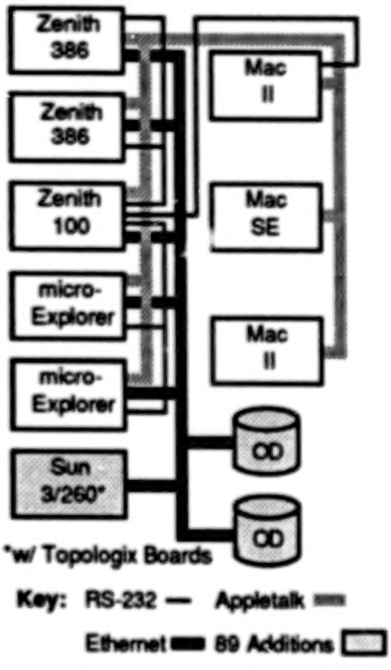


Figure 2. Mission Operations Laboratory

4.5.1. AI Microprocessing Environments[3]

Before 1988, microprocessor hardware capable of meeting current and projected requirements for AI computing did not exist. In 1988, Texas Instruments made a microprocessor system available that hinges on the recent development of technology allowing LISP processing on a single chip. The microExplorer™ provides most of the programming environment found in the parent Explorer machine and, similar to all dedicated LISP machines, is object-based. The microExplorer is hosted on a Macintosh II, with the microExplorer providing a dedicated LISP processing environment and the Macintosh II providing the I/O and user-interface. Functionally, the LISP processing environment is handled by the Macintosh as it would any other application. Indeed, other applications (e.g., Hypercard) can run simultaneously with the LISP processor. MicroNet™ and NuBus™ serve as the interface between the microExplorer and Macintosh II environments. Using either the microExplorer or the Macintosh II, capabilities exist for creating application programs that service the other processor. The Network File System™ (NFS) is an RPC server composed of several procedures and provides transparent file I/O capabilities for operating systems if requested. The eXternal Data Representation™ (XDR) protocol provides common data representation across networks.

4.5.2. Traditional Microprocessing Environments

As described earlier, future information management systems will be required to operate in environments where heterogeneous computing systems are the rule rather than the exception. The MOL provides for heterogeneous microprocessing capability by including, in addition to the microExplorers, traditional microprocessor facilities. Two Zenith 386s provide more standard OS/2 operating systems, and two standalone Macintosh IIs provide two different operating systems: one uses the standard Macintosh operating system and the other A/UX. Of course, both Macintosh IIs, as well as the microExplorers, provided extensive graphics capabilities. All MOL hardware is networked internally with the resulting LAN networked to the Information Network Laboratory.[4]

4.6. IIMS Prototype Development

Implementation of the IIMS prototype will proceed in a highly modularized fashion, with early iterations of functional modules representing simplified versions of later enhancements. Overall, we are planning for four phases of prototype development.

---

[2] The Topologix™ configuration may be dynamically reconfigured at runtime to emulate any of several parallel topologies. Additionally, hybrid architectures may be designed with one topology applied to one aspect of the application and a second topology being applied to another part of the application. Currently, parallel versions of FORTRAN, LISP, and C will run on the system.

[3] The Remote Procedural Call (RPC) and eXternal Data Representation (XDR) servers and the Network File System (NFS) are products of Sun Microsystems. NuBus and MicroNet are Products of Texas Instruments.

[4] The Martin Marietta I&CS Information Network Laboratory provides a flexible networking environment for proof-of-concept research, architecture design, and prototype development in the areas of tele- and data communications.

4.6.1. Phase 1 Development

Figure 3 shows our initial concept of projected information management functional allocation to software modules. During Phase 1, rudimentary versions of all functions will initially reside in one of the microExplorers. Briefly, the *User* function initially represents an abstract simulation of the 'typical' IIMS user; we anticipate several categories of potential IIMS users. However, while the prototype will examine functional support features for many of those categories, we will initially focus on the science user and support functions he will require. It should be apparent that many of the support features developed for one type of user may be generalized to other user types. Another important aspect of the *User* function is to allow remote access. Phase 1 *User* capability will not allow for remote access but will focus on the types of required user-support features. Two *Communication Interface* modules, one to handle communication between the *User* and the IIMS *Interface Manager* and the other which handles communication between the *Query Processor* and the *Database Manager*. The *Interface Manager* is the module which handles intelligent support features for the user. For example, information on user profiles and intelligent user views would be kept in this module, allowing for individual system configurations when the user logs on. An important feature of this component is allowing the user to modify his user view when appropriate, thereby 'modifying' the underlying *metadata* structure found in the *metadata*base (see section 4.3.1 for more detail). Additionally, all on-line *help* facilities would be included here, and the *User* would be able to create procedural command macros allowing for more efficient information navigation and access. The *Metadatabase Manager* interacts closely with both the *Interface Manager* and the *Query Processor* helping to direct the user to appropriate information without wasting an inordinate amount of the user's time. The *Metadatabase Manager* coordinates activity occurring in the *metadata*base which, it should be remembered, includes all information and knowledge about the system. Upon request from the *User*, navigation through the *metadata*base is controlled by the *Metadatabase Manager*, and is made more efficient by using user profile and views information. The *Query Processor* module intelligently decomposes and translates *User* queries, generating information requests to the science database. The *Database Manager* performs the functions of the typical database, performing search procedures when queries are received. The *Meta/Science Data Update System* intelligently extracts metainformation from incoming science data and determines initial links (i.e., potential navigation paths) and associations among the new and already existing data. As a result, the existing knowledge structure in the *metadata*base is modified to include the new information. Original science data, now with appropriate *metadata* linking information attached, is then stored in the science database.

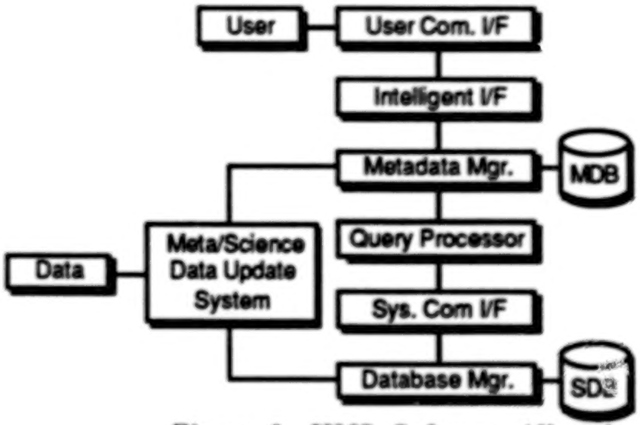


Figure 3. IIMS Software Allocation

A second major part of Phase 1 prototype development, is that part of the implementation which focuses on designing standardized message passing protocols and data structures, allowing users operating any hardware or software to interact with the IIMS as well as allowing external DBs to update IIMS information in a consistent and standard fashion. Several approaches to developing automated data information update capability in the *Meta/Science Data Update System* are possible. The conventional approach would be to first define all existing and projected data and then build a knowledge structure based on the contents and characteristics of those data. The result would be a knowledge structure analogous to a card catalog in a library. Such an approach presents several long-term problems: automatic information updates and general knowledge structure operations would be possible only within the bounds of the predefined data. As a user develops new ways to use existing instruments or analyze raw data, or when new instruments are deployed, new types of data will be produced. Modifications to the existing knowledge structure would have to be manually implemented. Also, it is highly unlikely that PIs will have the time or inclination to research and cross reference characteristics of new data to those of seemingly unrelated disciplines or problems. The capability and effective usefulness of the *Meta/Science Data Update System* would be limited to the contextual boundaries set by the user based on personal knowledge. This approach would likely provide an acceptable solution for the near-term, but it does not address long-term requirements. Standard data definition approaches do not provide the information needed for this level of intelligent operation.

To provide the types of information to make this level of capability possible, new techniques for describing data are needed. It is important to understand that to implement such a capability, not only data contents (traditionally described in a Data Description Language), but data characteristics, in a form suitable for cross referencing to other data types, must be addressed. One possible solution would be a structure which we term a *Data Set Descriptor* (DSD). A properly developed DSD would describe data in such a way that relational cross references can be made to characteristics of other data types. This would also give the *Meta/Science Data Update System* the capability of automatically creating new data categories and adjusting the knowledge structure based solely on the DSD input for a new data type. By having properly defined characteristics of this new data set, the *Meta/Science Data Update System* would be able to determine if the new data fit the existing knowledge structure or if the knowledge structure should be modified. The *Meta/Science Data Update System* would also be able to establish inferences required to recognize that new data may be useful for other, peripherally related investigations. Implementation of a DSD type data structure provides the foundation necessary for smooth system evolution. As new advances are made in AI and data representation techniques, the DSD provides the basis for integrating advances without massive system restructuring. An additional advantage to this approach would be the virtual elimination of software changes in the system to maintain data compatibility.

4.6.2. Phases 2-4

Primary modifications to the system developed during Phase 1 include the implementation of a true 'remote' user capability, the implementation of user communications control in a separate piece of hardware (Zenith 100[5]), the implementation of an external DBMS on a separate piece of hardware (the second microExplorer), and initial enhancements of several skeleton functions initially developed during Phase 1 (e.g., adding more support features in the intelligent interface module, adding more advanced query processing capability). While some functional enhancements will be added to the system, the primary objective during Phase 2 is to distribute functionality of the system developed during Phase 1 to other networked hardware components. The primary objective during Phase 3 is to significantly enhance the capabilities of both the *Intelligent Interface* module, the *Meta/Science Data Update System*, and the *Query Processor*, in addition to expanding the number and types of potential DBMSs with which the IIMS might have to interface (including the incorporation of an existing science database). Anticipated enhancements include the capability to handle multiple users with text or graphics-oriented user-interfaces, and the addition of an automated planning and scheduling capability that will allow resolution of user data request conflicts. Currently, it is anticipated that enhancements to the system during Phase 4 will include interfacing with X WINDOWS, increasing the number of distributed databases, providing on-line user data path switching, and simulating data capture, allowing for testing of alternative IIMS processing and storage topologies. To date, our work has focused on several IIMS areas and initial results of our prototyping work are briefly described in section 5.

5. RESULTS AND CONCLUSIONS

IIMS Operations Concept. We have completed a detailed examination of potential operations and associated functional requirements for a Space Station-era IIMS. This analysis served as the focal point for our prototyping work.

Data Modeling. We have been actively pursuing information concerning potential types of scientific data with which the advanced IIMS will have to be familiar. Discussions we have had with National Center for Atmospheric Research personnel, as well as documentation concerning typical satellite data related to atmospheric studies and sample data they have provided, have allowed us to more accurately define potential types of data and queries used to access that data. The sample data we have obtained includes data at a minimum of two processing levels and will include data that may be displayed in a graphical format. It is our intention to use the sample data, not only to provide insight into real scientific data structures but also as drivers for graphic displays that will likely be part of the IIMS's functional repertoire (e.g., browse data, quick-look data). We have also met with Earth Resources Observation System (EROS) data center personnel to discuss their data storage, access, handling, and distribution requirements and have received documentation concerning the types of data and database technology currently in place at the EROS data center. Information gathered from these discussions has helped focus prototype activity in several areas (e.g., user-interface requirements, accessing protocols, distributed database management and handling).

We have begun design of the module that will handle update messages received by the IIMS from heterogeneous data sources, and analysis of potential technical difficulties is proceeding. One issue currently being addressed is the format of update messages. Because of differences in types of scientific data (e.g., image, text, numerical) and their corresponding attributes, a variable format system is being analyzed for suitability to this environment. Problems now being studied include how to allow (1) automatic generation of these variable update messages, and (2) interactive generation of these messages in which the software handles proper message formatting depending, in addition to several other factors, upon the data type. We have also begun to analyze how *metadata* should be stored in the IIMS database, and the manner in which intelligent support features can be incorporated to allow efficient access to data. Technical issues involve both the representation of the knowledge and its automatic generation and incorporation into the system. Initial results indicate an object-oriented data representation technique to be the most flexible given projected functional requirements the *metadata* must support.

---

[5] The Zenith 100 has unique communications capability with 32 available ports.

User-system Interface. An analysis of NASA's Space Station Information System human-computer interface requirements has led to the initial prototyping of one potential IIMS user-interface concept. The user-interface is interactive and dynamic as well as both graphic and text oriented, allowing for multiple screen configurations suited to different IIMS functions.

Catalog Management. The *Data Management System* (DMS) *Tool* has been developed to provide data management capabilities for the TI microExplorer. It is based on the relational data model with a few extensions to allow for more flexibility. The system provides both a *Data Definition Language* (DDL) and a *Data Manipulation Language* (DML). The DDL allows for the definition, destruction, saving, loading and structural modification (adding, deleting and modifying of relational attributes) of databases and relations. The DML provides for an unrestricted query capability and relational set operations (join, intersection, union and difference). The query capability allows the user to provide any s-expression (LISP expression) as the selection criteria and project any combination of attributes from the result. The system provides for simple concurrency control through a series of locks and tokens on the database and relational levels (single modifier, multiple concurrent access). Multiple databases can be active at one time; however, querying between databases is not yet supported. Examples of DBMS functions that are not currently implemented are query optimization, transaction processing, index capability, crash recovery, and other advanced DBMS capabilities. The DMS will initially be used to implement the IIMS database and handle other IIMS prototype database needs. It has already been used to demonstrate how dynamic menus of IIMS database items can be generated from an IIMS database.

Interactive Graphics Objects (IGO). This system was developed to provide for dynamic binding of interactive command structures to mouse-sensitive graphical objects. The user can build command tables consisting of actions that result from mouse button activation; documentation for the appropriate activity is displayed whenever the mouse is over the object. Various combinations of commands can become "command modes" which can then be bound to different graphical objects; these IGOs can be manipulated just like any other graphics object on the microExplorer, and when the mouse is positioned over one of these objects, the specified "command mode" will become active. IGOs can be used to provide a wide range of capabilities. Because of the general capability which this system provides, it has been used to implement an interactive graphical display of the catalog hierarchy and an initial interactive tutorial system that simulates IIMS functionality (including animation and interactive *help*).

Communications Protocol and Message Passing System. An initial message passing/communications protocol format has been established and will be tested this year in terms of remote user access and automated data updates to the IIMS. Instrumental in the development of this protocol was the the implementation of a communications program providing direct user communications and querying capability between any combination of nodes on the MOL AppleTalk network. Also completed were network monitor and control programs. Current development efforts are focused on understanding and using the Macintosh RPC protocol. To date, we have successfully implemented a low level data transfer function which consists of each system requesting and receiving the current system time from the other. We are currently working to expand the function's capability and provide interfaces to Macintosh system applications.

Query and Browse Facility. Several topics related to IMS-querying capability have been studied and progress has been made on prototype functional implementation in some of these areas. Topics of focus have included (1) types of queries which will need to be handled by the IIMS, (2) identifying and prototyping special requirements necessary for handling spatial queries, (3) query formulation, and (4) the structure of the associated knowledge base(s). We will use several techniques to aid the IIMS-user in the location of information. Some of the techniques we have begun to analyze are constrained query formulation (including spatial and temporal selection), conceptual views (aided by user profiles), browsing, knowledge base assistance, cooperative response, query reformulation, suggestions of related data, and various forms of on-line *help*. A second issue we have begun to study is that of integrating browsing and querying facilities for the IIMS. This work is in the early stages of analysis, but we currently expect that browsing will probably provide the user with the fastest method for dynamically limiting the scope of the data for a particular query, for which no view is defined. As the user browses through the data hierarchy, the system will place restrictions on any query formed. Much work needs to be done on the definition of a database structure that will allow these two methods to be smoothly integrated. Current prototype implementation includes a constrained query formulation capability along with allowing the user to graphically select spatial information concerning data of interest. The user can browse IIMS *metadata* through either a menu or graphical representation.

Tutorial and User profile Capabilities. Both of these functions are in the rudimentary stage with a basic user profile capability being established. Currently, user profiles may be modified through either an IIMS operator's screen or through the user's own screen. Since objects are used as the representation technique for users, modifications to profiles are easily made. Future work will begin to tie user profiles to querying and browsing capabilities. Extensive use of the

IGO system will be made to develop the prototype on-line tutorial system.

Work described in this paper suggests several conclusions: First, concerning our approach, use of rapid prototyping and AI programming environments appears to be an efficient and cost-effective means of testing several proposed requirements for projected advanced ground systems. Second, our work in advanced information management and access has given us the springboard to test more sophisticated IIMS concepts in a highly modular and dynamic environment. The working prototype we have developed has already demonstrated the feasibility of applying object-oriented programming and advanced database concepts to problems associated with projected Space Station-era information management systems. Future work will continue to extend the current IIMS prototype by testing several advanced IIMS features, particularly in the area of remote, distributed access.

REFERENCES

1. Blaha, M. R., Premerlani, W. J., & Rumbaugh, J. E. (1988). *Relational Database Design Using an Object-oriented Methodology*. Communications of the ACM, 31(4), 414-427.

2. Campbell, W. J., Roelofs, L. H., & Short, N. M. (1987). *The Development of a Prototype Intelligent User Interface Subsystem for NASA's Scientific Database Systems*. NASA Technical Memorandum 87821.

3. Carnahan, R. S., & Mendler, A. P. (1987). *Rapid Prototyping and AI Programming Environments Applied to Payload Modeling*. Proceedings of the Third Conference on Artificial Intelligence for Space Applications, Huntsville, Alabama, 255-260.

4. Conklin, J. (1987). *Hypertext: An Introduction and Survey*. In I. Greif (Ed.), Computer-Supported Cooperative Work: A Book of Readings. Morgan Kaufmann, 423-475.

5. Hutchins, E. L., Hollan, J. D., & Norman, D. A. (1986). *Direct Manipulation Interfaces*. In D. A. Norman, & S. W. Draper (Eds.), User Centered System Design. Lawrence Erlbaum Associates, 87-124.

6. King, R. (1986). *A Database Management System Based on an Object-oriented Model*. In L. Kerschberg (Ed.), Expert Database Systems. Benjamin/Cummings, 443-468.

7. Li, Q., & McLeod, D. (1988). *Object Flavor Evolution Through Learning in an Object-oriented Database System*. Proceedings of the Second International Conference on Expert Database Systems, Tysons Corner, Virginia, 241-256.

8. McArthur, D. J., Klahr, Pl, & Narain, S. (1986). *ROSS: An Object-oriented Language for Constructing Simulations*. In P. Klahr, & D. Waterman (Eds.), Expert Systems: Techniques, Tools, and Applications. Addison-Wesley, 70-91.

9. NASA (1988). *A Program For Global Change - Earth System Science: A Closer View*. A Report of the Earth System Sciences Committee NASA Advisory Council.

10. NASA Goddard Space Flight Center (1986). *Earth Observing System Data and Information System Report of the Eos Data Panel - Volume IIa*. NASA Technical Memorandum 87777.

11. NASA Goddard Space Flight Center (1988). *Customer Data and Operations System (CDOS) Concept Definition Document*. NASA GSFC.

12. Norman, D. A. (1986). *Cognitive Engineering*. In D. A. Norman, & S. W. Draper (Eds.), User Centered System Design. Lawrence Erlbaum Associates, 31-61.

13. Polson, M. C., & Richardson, J. J. (Eds.) (1986). *Intelligent Tutoring Systems*. Lawrence Erlbaum Associates.

14. Short Jr, N. M., Campbell, W. J., Roelofs, L. H., & Wattawa, S. L. (1987). *The Crustal Dynamics Intelligent User Interface Anthology*. NASA Technical Memorandum 100693.

15. Short Jr, N. M., Wattawa, S. L. (1988). *The Second Generation Intelligent User Interface For The Crustal Dynamics Data Information System*. Proceedings of the 1988 Goddard Conference On Space Applications Of Artificial Intelligence, Greenbelt, Maryland, 313-327.

16. Stefik, M., & Bobrow, D. G. (1986). *Object-oriented Programming: Themes and Variations*. AI Magazine, 6(4), 40-62.

17. Wenger, E. (1988). *Artificial Intelligence and Tutoring Systems*. Morgan Kaufmann.

18. Zhu, J., & Maier, D. (1988). *Abstract Objects in an Object-oriented Data Model*. Proceedings of the Second International Conference on Expert Database Systems, Tysons Corner, Virginia, 3-16.

BLANK PAGE

An Intelligent User Interface for Browsing Satellite Data Catalogs

Robert F. Cromp and Sharon Crook
Science Applications Research, Inc.
National Space Science Data Center
NASA/Goddard Space Flight Center
Greenbelt, MD 20771

Abstract

A large scale domain-independent spatial data management expert system that serves as a front-end to databases containing spatial data is described. This system is unique for two reasons. First, it uses spatial search techniques to generate a list of all the primary keys that fall within a user's spatial constraints prior to invoking the database management system, thus substantially decreasing the amount of time required to answer a user's query. Second, a domain-independent query expert system uses a domain-specific rule base to preprocess the user's English query, effectively mapping a broad class of queries into a smaller subset that can be handled by a commercial natural language processing system.

The methods used by the spatial search module and the query expert system are explained, and the system architecture for the spatial data management expert system is described. The system is applied to data from the International Ultraviolet Explorer (IUE) satellite, and results are given.

1 Introduction

Large amounts of data from a variety of sources covering a multitude of domains are stored in databases. Increasingly, these databases are becoming more complex and intertwined. In addition, research involving this data often is multidisciplinary in nature, requiring the researcher (scientist or manager) to access multiple databases over a network of non-homogeneous computer systems. Users are investing their important time in learning and recalling how to use these systems. Unless a person has a pressing need to know an answer, a system is not learned—the start-up time for learning how to use the system outweighs the importance of the answer.

The Intelligent Data Management (IDM) Project at the National Space Science Data Center (NSSDC), NASA/Goddard Space Flight Center, has been involved over the past few years in investigating, developing, and proving methods which facilitate the accessing of databases for the user [4, 15]. Our research has concentrated on combining specially developed software with commercial packages, and applying our systems to non-trivial domains.

The general domain-independent methodologies which the IDM research has produced define the steps that are necessary for automatically developing an intelligent user interface to access databases. This paper reports on our approach to creating a spatial data management expert system that serves as an intelligent front-end to databases that contain spatial data. We believe this is an important step in designing a system that researchers can use to access the voluminous amounts of data that will be produced by on-going and future NASA missions, such as IUE, Space Station, Space Telescope and Eos.

2 The Spatial Data Management Expert System Architecture

We have designed and implemented a large scale domain-independent spatial data management expert system that provides answers to users' English queries. The system integrates our in-house developed modules with various commercial products.

Figure 2.1 shows the conceptual arrangement and data flow of the system. A user enters his query in English, and optionally, by means of a graphics interface, selects a set of spatial regions to which he wants to restrict his query. The English portion of the query is passed through a query expert system that uses a domain-specific rule base to preprocess the query for handling by DataTalker, a commercial natural language database front-end marketed by Natural Language, Incorporated [13]. The query expert system channels a broad class of queries into a smaller set which DataTalker is then able to process. In addition, the query expert system passes the region coordinates to a specially designed spatial search module that returns a list of the primary keys of the records in the database that fall within the selected regions. DataTalker is passed an English query that is formed from the user's transformed query and the set of primary keys.

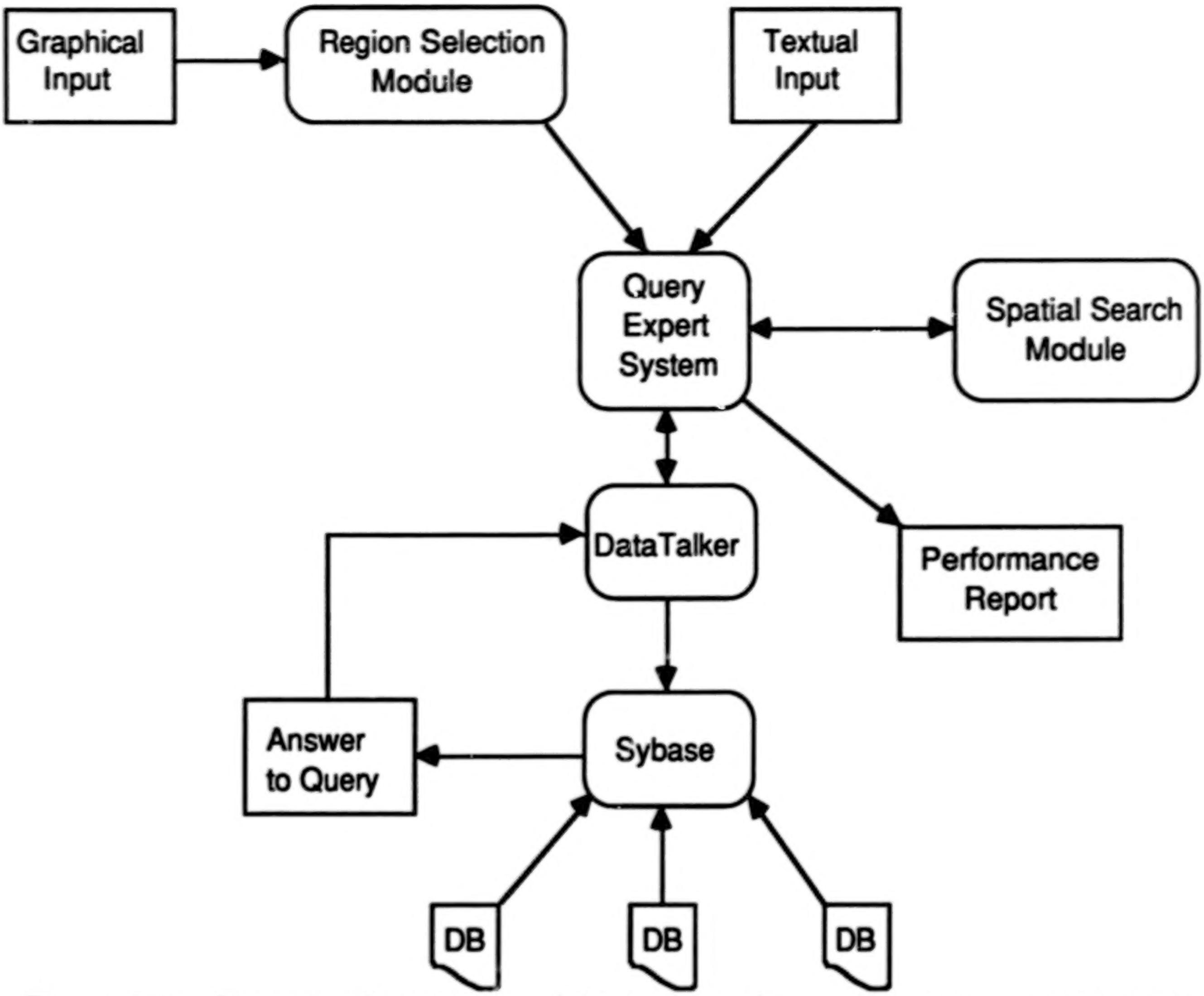

Graphical Input
Region Selection Module
Textual Input
Query Expert System
Spatial Search Module
DataTalker
Performance Report
Answer to Query
Sybase
DB
DB
DB

Figure 2.1: The overall system architecture for the spatial data management expert system.

When DataTalker is installed on a machine, it is connected to a database management system. DataTalker automatically invokes the DBMS whenever it successfully parses a transformed query. The low-level database accessing is transparent to the user, and the format of the query results is controlled by DataTalker. If DataTalker cannot parse the query (see Section 4), it returns a message to the query expert system indicating so.

The Spatial Data Management Expert System causes DataTalker to store its result in a file. The contents of this file are used to update a text window of query results on the user's console.

If the user is not satisfied with the way his query is transformed or answered, he can file an electronic performance report which is later reviewed by the domain expert. The domain expert uses this information to guide the evolution of the system with respect to the types of queries asked by the users.

3 Spatial Search

In order to answer queries about spatial data, we must be able to access the information associated with those records quickly. A database is too slow for this problem forcing the development of quicker methods for accessing data. We chose quad trees and k-d trees as suitable data structures for answering spatial queries about records. The implementation of these data structures allows us to perform such tasks as finding the nearest neighbor to a given coordinate and determining all records within a certain region. After finding the answer to such spatial queries, we can return the appropriate primary database keys to the expert system which uses the keys to gain any other necessary information from the database.

3.1 Tree Construction

Our implementation of these two data structures relies on the spatial relationships among records based on coordinate values for a two dimensional space, for example right ascension and declination. We use similar methods to build each of these two types of trees from a set of n records, $\{x_1, x_2, \ldots, x_n\}$. In order to create a quad tree, we choose a point, x_i, from this set. This point determines the root node of the tree and divides the space into four quadrants based on the coordinate values of that point. Each quadrant is represented by one of the four subtrees extending from the four children of x_i. We continue to divide the space in this manner until each record is represented by a node in the quad tree. Figure 3.1 shows an example of the construction of a quad tree from the data in Table 3.1.

Table 3.1: Sample observation data (simplified).

Name	Right Ascension	Declination
Algol	3	41
Altair	20	9
Bellatrix	5	6
Edasich	15	59
Regulus	10	12
Sheratan	2	21

The creation of the k-d tree differs slightly in that the point chosen from the set of records is used to divide the space into two subspaces which determine two subsets of the set. This process continues in alternating dimensions until each record is represented by a node in the k-d tree. Figure 3.2 depicts this process.

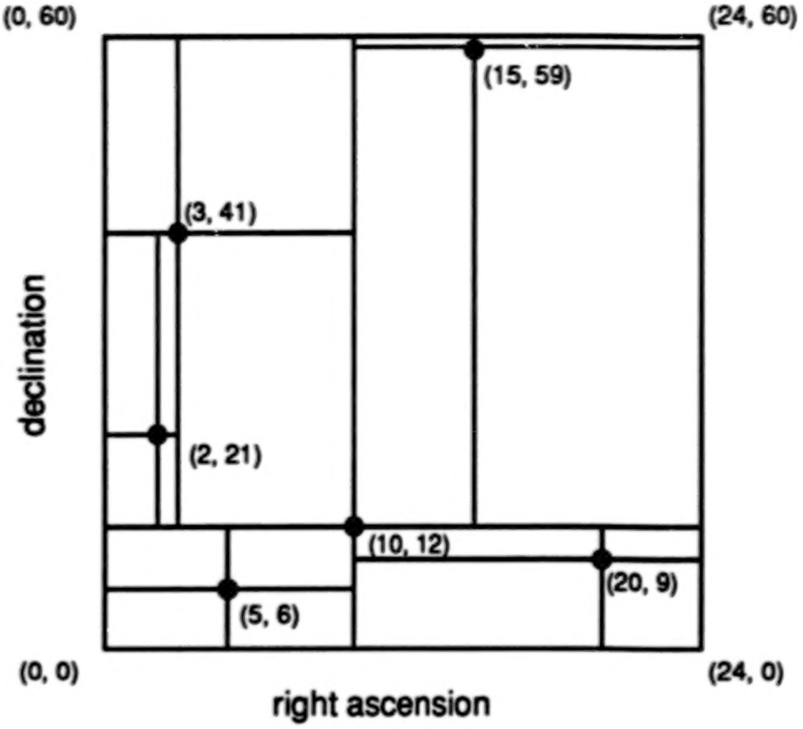


Figure 3.1: The structure of a quad tree.

We chose to store the IUE observation data in the internal nodes of the trees due to the large number of records and the fact that deletions are not needed for our application. If node deletions had been necessary, we would have chosen pseudo quad trees and pseudo k-d trees for implementation. These structures store data associated with points only at the leaf nodes of the tree providing the means for a large number of efficient deletions and insertions [14]. An observation catalog continues to grow as long as the satellite which supplies its data continues to function. For this reason we needed the capability to insert a small number of observations after the completion of the trees, which is possible and efficient at the leaf level if we allow a slight depth increase.

Due to the nature of these data structures, we developed an object oriented implementation in C++ [16]. The key structure in C++ is a user-defined type called a class. In our application classes include such objects as nodes, lists of nodes, trees, and regions in space. This object oriented approach is particularly well-suited to this problem due to the hierarchical nature of these structures.

Another advantage of such an approach is the ability to create programs which are concise, easy to understand, and highly maintainable.

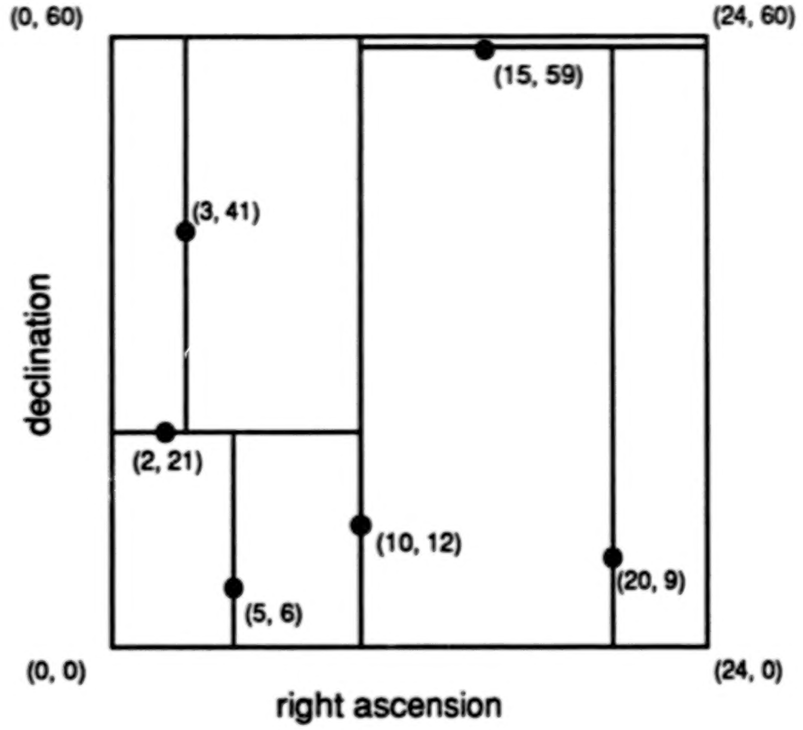


Figure 3.2: The structure of a k-d tree.

In order to minimize search times for the quad tree and k-d tree, we chose to expend more effort in building the trees. By allowing more time for preprocessing, we can optimize the trees through balancing. This is accomplished by imposing the condition that for every node x_i in the tree, every subtree of that node may contain no more than half of the nodes whose root is x_i. This is easily done by choosing the proper point in each observation subset when placing nodes in the tree. At each step in the recursive process we must select the median point in either dimension. Thus all remaining observations in that subset will be equally divided with respect to the point chosen meeting the condition that no subtree of the new node may contain more than half the nodes in that subset. This optimization insures that the maximum path length in a tree of n nodes is the floor function of ($\log_2$ n) and the maximum total path length is $\sum_i \text{floor}(\log_2 i)$ [6].

There are various quick algorithms for selecting the kth element of a set of n elements and partitioning the set about that element. These provide efficient methods of determining a median and implementing the process described above. One algorithm due to C. A. R. Hoare (referenced in [2]) performs the task in O(n) average time and is easy to implement. This allows us to create an optimized tree in $O(n\log_2 n)$ time. We chose this algorithm for its simplicity and efficiency. Another more complicated algorithm due to Floyd and Rivest uses only n+k+o(n) comparisons and runs very quickly [7].

When we wish to create a tree, we use the standard C++ **new** command to allocate a large block of virtual memory which will store the tree nodes. After creating the tree from the data in an observation catalog, we can send all of the information needed for rebuilding the tree to a file. This information includes the right ascension and declination values, the primary keys of the database relation, and the addresses of the children for each node in the tree. We subtract the base-address from these addresses in order to save the general tree structure which is independent of its location in memory. We can rebuild the tree using this data by adding the node address values to the base-address for a newly allocated block of memory. With this ability, the preprocessing time necessary for initially creating and balancing a tree is eliminated. Trees are always readily available and efficiently recreated.

3.2 Spatial Queries

The simplest type of query which can be answered by searching a quad tree or a k-d tree is what Knuth calls a "simple query" [10]. This involves searching a tree for a specific record using a point search based on the spatial data. A simple algorithm performs the search by making a comparison at each node and choosing the correct subtree for the next test. The average time required is proportional to the total path length divided by the number of nodes in the tree [6]. Thus no simple search will require more comparisons than the maximal path length.

A more general type of spatial query is a "region query" [10] in which we specify a set with which records must intersect. This class of query might be invoked by a question such as "List the primary keys of all observations with right ascensions between 8 hours and 16 hours and declinations between 20 degrees and 50 degrees." Figure 3.3 shows an example of such a region in space.

The conventional method of answering a rectangular region query by searching a quad tree involves a recursive procedure [6]. An almost identical algorithm will suffice for k-d trees [1]. Similar procedures can be defined for different types of geometric figures such as the shape of a satellite camera aperture.

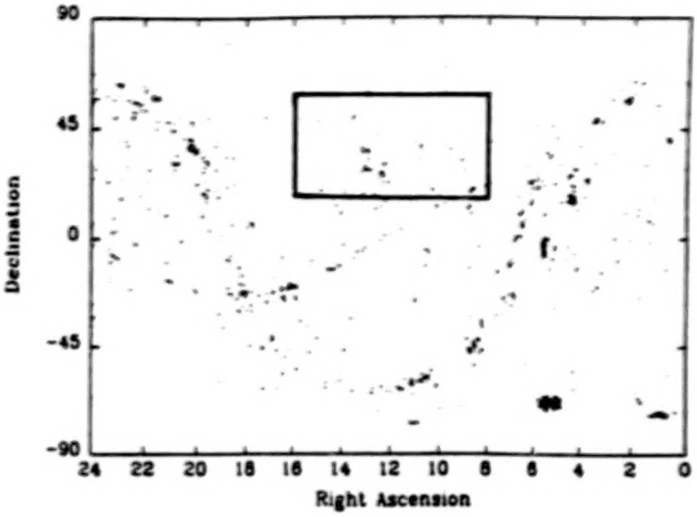


Figure 3.3: An example of a region to be searched.

The last type of query necessary for our application is a nearest neighbor query. Given spatial coordinate values, it is often desirable to find the nearest observation to that point. Bentley provides a complicated algorithm for performing this process in k-d trees for which empirical tests show $O(\log_2 n)$ time [1]. A slight variation due to Friedman, Bentley, and Finkel was *proven* to be $O(\log_2 n)$ [8]. For simplicity, we implemented a more naive algorithm for finding a nearest neighbor in either type of tree which relies on the region search described above. Future empirical tests are planned in order to determine the performance of our algorithm.

4 Natural Language Processing

There are a small number of commercially available natural language processing packages. We have found DataTalker, by Natural Language Incorporated, to be a very powerful system for handling English queries. DataTalker can communicate with most of the major database management systems. The user's query is converted into the necessary database query language, the DBMS is invoked, and DataTalker displays the query result to the user in a structured format. If DataTalker is unable to parse the user's query, a message is generated explaining why. A query that is syntactically incorrect, contains metaphorical expressions, or consists of specialized vocabulary which DataTalker has not been taught causes an error message to be produced.

BEST COPY AVAILABLE

DataTalker must be configured to the domain. The person who does this configuration must be familiar with both the domain and the structure and contents of the database. We refer to this person as the domain expert. Preferably, this person will have a list of questions that the user base might ask.

Since we have had limited experience with DataTalker (this was our first use), we aren't qualified to report on the amount of time it takes to configure the system for some domain. However, we will note that the current version of DataTalker (version 3.0) is much friendlier than its previous versions, and we find that it is easy to build a configuration in incremental steps.

5 The Query Expert System

DataTalker was designed to be used solely interactively. This is unfortunate, because many different systems could benefit from its capabilities if an interface existed that allowed it to be included as a component. We found a natural language interface was desirable for many of the domains with which we were working, so we faced the choice of either circumventing DataTalker's limitations, or designing our own natural language processing system. We quickly adopted the former option, since the latter requires a long-term effort into computational linguistics if a robust, mature system is sought [9,11].

If a person submits a query to DataTalker that the system is unable to understand, a short message is returned which may or may not indicate the problem. The person is given immediate feedback and is expected to rephrase his query so that the system can answer it, or abandon it because it is obviously beyond the scope of DataTalker. If the system is able to parse the query, render it in the necessary database query language, and successfully query the database, then the result is returned to the user. In any case, the person quickly realizes whether his query was handled properly.

When DataTalker realizes that it is unable to handle a query, it returns a brief message indicating the problem, possibly preceded by its English interpretation of the query. The message typically starts with a phrase such as "Sorry, ... ," "The database contains no information on ... ," or "I don't know ... ."

There is also a possibility that DataTalker misinterprets the user's query and provides what it believes to be a correct answer, although in reality the answer corresponds to a question different from the one the user asked.

The query expert system (QES) is a module we designed and implemented in LISP which serves as an intelligent front-end to DataTalker. It accepts the user's query, transforms it according to a domain-specific rule base, and passes it on to DataTalker. If DataTalker replies that it does not understand the query, then QES packages the original query along with the transformations it underwent and DataTalker's response, and files this. The domain expert periodically reviews this file, and makes changes to the rule base, if possible, so that QES can transform the problematic queries into queries that can be parsed by DataTalker. This feature allows the system's performance to improve throughout its lifetime.

5.1 The Textual Rule Base

The domain expert creates a text file that contains the rules and functions which are used to transform the user's query prior to passing it to DataTalker. The grammar for representing the textual rule base is shown in Figure 5.1.

```
<rule base> ::= <rule> <rule base> | <function> <rule base> | ε
<rule> ::= <pattern> '==>' <transform> {<explanation>}
<pattern> ::= <pattern head> <pattern tail>
<pattern head> ::= <word> | <number matchup> | <word set>
<pattern tail> ::= <pattern item> <pattern tail> | ε
<pattern item> ::= '...' | <word> | <number matchup> | <word set>
<word> ::= <ascii character excluding whitespace>+
<number matchup> ::= '#'{<digit>}*
<word set> ::= '{' <word> <word list> '}'
<word list> ::= ',' <word> <word list> | ε
<transform> ::= <transform item> <transform> | <transform item>
<transform item> ::= <word> | <function call>
<function call> ::= <function name> '(' <parameter list> ')'
<function name> ::= <letter> <ascii character excluding whitespace>+
<parameter list> ::= <number matchup> <number matchup list>
<number matchup list> ::= ',' <number matchup> <number matchup list> | ε
<function> ::= <function call> '=' { <function body> | '(' <expression> ')' }
<function body> ::= { <condition> ',' <action> <carriage return> }+ <blank line>+
<condition> ::= <simple comparison> | <condition> 'and' <condition>
<simple comparison> ::= <operand> <relational> <operand> { <relational> <operand> }
<operand> ::= <number matchup> | <number>
<relational> ::= '<' | '>' | '=' | '<=' | '>=' | '<>'
<action> ::= '!' | { <word> | '(' <expression> ')' }+
<expression> ::= <mathematical expression possibly involving parameters to function>
```

Figure 5.1: The grammar for defining domain-specific rules and functions.

If DataTalker is unable to handle a query in one form, but can successfully answer the same query when it is reworded, then one or more rules can be added to the rule base so that QES can automatically transform the misunderstood query prior to passing it to DataTalker. Application of a rule transformation causes a portion of the query to be restructured, deleted or replaced by text specified by the rule. QES continually applies its domain rules to the successive renderings of the query until no rules are activated. The final form of the query at this point is displayed to the user along with explanations of why various transformations were used. The user must then consent that the altered version of the query is the closest match between his original query and the information directly obtainable from the database, or else the query is not sent to DataTalker. If the user is unsatisfied with the way his query has been altered, then he is allowed to file an electronic report stating his reasons. This, along with the QES actions for the query, can later be reviewed by the domain expert leading to possible enhancements to the rule base, or clarification back to the user.

A query transformation rule consists of a pattern and the transform which is to occur given that the pattern is encountered in the query. An optional explanation can also be supplied with the rule. This explanation is displayed to the user if this rule is used to alter his query.

A pattern is a series of pattern items. The possible pattern items are ..., <*word*>, <*number matchup*>, and <*word set*>. The pattern item ... matches zero or more consecutive words of the query. The item <*word*> must match the query word exactly. A <*number matchup*> matches the query word if it is a number. A <*word set*> is a list of words, and a match occurs if any member of this set is the same as the query word. An implicit ... starts and ends each pattern. Each word in the query must be accounted for by some item of the pattern in order for the pattern to match the query. In addition, if item *i* directly precedes item *j*, then the portion of the query matched by item *i* must directly precede the portion of the query matched by item *j*. Figure 5.2 shows the match between a query and a pattern.

QUERY: show all the observations taken at the 2175 A bump
PATTERN: ... observations ... {at, of} the # A bump ...

Figure 5.2: The match-up between a query and a pattern.

In a rule, the transform describes how the query should be altered provided the pattern matched it. A transform consists of a combination of one or more function invocations or words. The substitutions in the query that occur depend on the make-up of the transform. The ordering of words in the new query is based on the pattern. All ... pattern items are incorporated unaltered into the new form of the query. The left-most <*word*> pattern item that also occurs in the transform is replaced by the entire transform in the new query. If the transform consists solely of a function invocation, then the left-most number in the pattern that is bound to one of the function's parameters is replaced by the value returned when the function is evaluated. All other query words and numbers that are matched to <*word*>, <*word set*>, or <*number matchup*> pattern items are deleted. Figure 5.3 shows the effects of a rule firing on a query.

QUERY: how many observations were made at high resolution
RULE: observations ... {at, with} high resolution ==> high dispersion observations

Pattern Item	Query Word	Contribution to New Query
...	how many	how many
observations	observations	high dispersion observations
...	were made	were made
{at, with}	at	
high	high	
resolution	resolution	
...	ε	

NEW QUERY: how many high dispersion observations were made

Figure 5.3: How a rule alters the structure of a query.

The domain expert can also define functions that can be used in the rule's transform. Figure 5.4 is one example of a function and a rule that uses that function. A function definition consists of the function name, a list of its parameters, and the function body. The body can be either a mathematical expression or a collection of condition-action pairs. A function that is defined as a straight mathematical function returns the result of evaluating that function under the current function parameter bindings. If the body consists of condition-action pairings, then if a condition is found that holds true for the given parameter bindings, the action is substituted in place of the function call in the rule's transform. If an action has an expression embedded within it, that part is replaced by its evaluated form. The cut action "!" has special significance. Conditions whose action is "!" are evaluated prior to all other conditions. If a condition holds whose action is "!", the rule which invoked this function is not applied for the current situation. Figure 5.5 shows two rules that could both potentially fire for a given query, even though their results are quite different. The cut ensures that the correct rule is applied.

When a <*number matchup*> pattern item matches a number in the query, a binding occurs such that all occurrences of that <*number matchup*> token that appear in the transform are replaced by the number it matched from the query. If more

```
RULE:  observations ... at the # A bump ==> f(#) observations

f(#)   =  1000 <= # < 1920, short wavelength
          2000 < #, long wavelength
          1920 <= # <= 2000, short wavelength or long wavelength
          # < 1000, !
```

Figure 5.4: An example of a rule which relies on an expert-defined function. The rule does not apply if the *<number matchup>* token is bound to a value less than 1000.

than one *<number matchup>* item occurs in a rule's pattern, the *<number matchup>* tokens should be distinct from one another. If the expert wants to write a rule that fires only if, say, the same number appears twice in the query, then the pattern should use two different *<number matchup>* tokens, and the transform should invoke some function which checks that the bindings of the two tokens are equal in value.

```
RULE 1:  observations ... at # A ==> f(#) observations
RULE 2:  observations ... at # A ==> g(#) observations

f(#)   =   1000 <= # < 1920, short wavelength
           2000 < #, long wavelength
           1920 <= # <= 2000, short wavelength or long wavelength
           # < 1000, !

g(#)   =   # < 4, high dispersion
           # >= 4, low dispersion
           # > 10, !
```

Query	Transformed Query (Rule used)
show the observations taken at 1 A	show the high dispersion observations (2)
show the observations taken at 1800 A	show the short wavelength observations (1)

Figure 5.5: Use of a cut "!" can ensure the correct application of a rule.

5.2 Compiling the Rule Base

The textual rule base must be compiled before QES can use it. We have developed a LISP module that parses and compiles the expert's rule base. Compilation must occur any time the textual rule base is altered if QES is to make use of these changes. Four structures are computed during compilation: the rules, the functions, the word occurrences, and the rule thresholds.

Each rule is encoded as a list consisting of a unique rule number, the pattern and the transform. The pattern is represented as a list of the pattern items, in string format, where the *<word set>* pattern item is structured as a list of strings. The transform is rendered as a list of its components. A function call is stored as a list whose head element is the function name and whose remaining elements are the parameters to be passed.

Each function is represented as a list consisting of the function name, a list of its parameters, and an evaluatable LISP encoding of the function body. If necessary, the condition-action pairings are reordered so that the potential cuts are evaluated prior to the other clauses. If a mathematical expression occurs in the function body, then code is generated which converts the result to a string automatically so that it can be concatenated into the action or template correctly. Figure 5.6 shows the encoding for one of the functions from Figure 5.5.

```
("f" ("#")
    (cond  ((< "#" 1000) "!")
           ((and (<= 1000 "#") (< "#" 1920)) "short wavelength")
           ((< 2000 "#") "long wavelength")
           ((and (<= 1920 "#") (<= "#" 2000)) "short wavelength or long wavelength")))
```

Figure 5.6: The compiled form of an expert-supplied function. The function body can be evaluated once the value for the parameter is substituted.

A word occurrence index is also computed and placed in the compiled rule base. An element of this index consists of a word and a list of those rules in which that word appears, and the number of times it appears in each of those rules (only the rule's pattern is used, not its transform). Instead of indexing specific numbers, the element "#" appears in the index along with a list of the rules which contain numbers and the number of occurrences in each rule. An entry is made for each word in a <*word set*> pattern item, whereas the ... pattern item is ignored. The word occurrence index is used during run-time to increase the speed of the inference engine.

The last structure that is placed in the compiled rule base is referred to as the rule threshold list. An element of this list consists of a rule number and the number of items in that rule's pattern, excluding ... pattern items. This list is also used by the inference engine in its attempt to find the correct rule to apply.

5.3 The Inference Engine: Applying the Rule Base

The QES inference engine uses the query to decide which rules are potentially applicable for transforming it. The method of using the data to drive the inferencing is referred to as forward chaining.

The size of the rule set is most likely much larger than the number of words in the user's query. The user would experience a noticeable wait if the inference engine had to inspect each rule every time it was seeking to transform the query. Also note that there are relatively few rules applicable to the query at any given time. The inference engine must be implemented so that it minimizes the number of rules it attempts to apply for a given query.

Each rule starts with a score of zero. A list is formed of the unique words that are found in the query and their number of occurrences. Each word in turn is taken from this list, and is used with the occurrence index to find those rules in which it occurs. The score for each of these rules is incremented by one for each time the word occurs in its pattern, not exceeding the number of times the word occurs in the query.

All rules whose score equals or exceeds their rule threshold value are placed in the conflict set. These are the rules that can potentially affect the given query at this time. All the other rules cannot apply, either because the query is shorter than their patterns, or their patterns contain one or more words which are not in the query. The rules found in the conflict set may or may not apply. Even though their patterns contain the appropriate words, the words could be in a different order, or a function invocation in their transform could return a cut "!".

More than one rule in the conflict set may satisfy all the conditions for firing. However, one rule may fit the circumstances better than another because it is more specific [3]. To ensure that the "best" rule is found and applied as quickly as possible, the rules in the conflict set are ordered according to their specificity. A rule is considered more specific than another if its pattern contains more items. Rules with longer patterns are tried before ones with shorter patterns. A rule is automatically removed from the conflict set if it is unsuccessfully applied. Once a rule is found that can fire, a new query exists, and the conflict set must be recomputed anew.

In general, a given query undergoes a number of transformations until the inference engine is unable to find any rules which can alter it. This is known as a quiescent state and corresponds to an empty conflict set. Once this state is achieved, the final form of the query is returned, it is displayed to the user, and QES passes it to DataTalker. We must make sure that a quiescent state is reached. In particular, we must guard against a collection of rules which bounces the query back-and-forth. To remove the possibility of infinite looping, the principle of refraction [12] is used. A rule is removed from the conflict set if it has already been applied to the query in its current form. Most likely, if infinite recursion accidentally was introduced to a rule base, only a handful of rules would be involved. This method of detecting infinite looping works for any number of rules.

6 Results

To test our spatial data management expert system, we have selected a subset (about 5,000 records) of the IUE Minilog (about 70,000 records). After interviewing an astrophysicist who is an IUE expert, we developed a LISP program that converted the magnetic IUE Minilog tape data into a file consisting of the various fields available (e.g., right ascension) or computable (e.g., continuum level) for each observation.[1] The resulting file of IUE observations was ingested into Sybase, a commercial DBMS available from Sybase, Inc. [17].

DataTalker was configured as far as possible on the IUE domain, and a rule base for QES was generated which contains about 50 rules, some of which have been used as examples throughout this paper. Figure 6.1 contains a list of some of the questions our system can answer, where italics indicates that QES performs some modification to that part of the query before it is passed to DataTalker.

What observations *of HD112244 have been taken*?
What high dispersion IUE observations *of HR4908 have been taken*?
What observations *around 1260 A* have been taken of CPD -56 5498?
What observations of O-stars have been made *at 0.1 A resolution at 2600 A*?
What low dispersion observations have background levels *larger then* [sic]*100 DN* and continum [sic] levels less than *220 DN*?
What low dispersion observations have been made *from 1150 - 3000 A*?

Figure 6.1: Actual user queries that can be successfully answered by our spatial data management expert system. Italicized portions of the query are modified by QES prior to being passed to DataTalker.

By using our region selection and spatial search modules, any of the questions can be restricted to a smaller portion of the database. We believe this increases the speed of the system because our method of performing the spatial search is significantly faster than relational database search methods that use non-spatial primary keys to construct their trees. The quad and k-d trees take full advantage of the numerical properties inherent in the spatial data, whereas the trees constructed by DBMS's must be searched entirely to retrieve all records which meet some spatial criteria. Our spatial data management expert system presents a list of the primary keys to the DBMS immediately, thus eliminating the need for the database to be exhaustively searched.

---

[1] We are currently developing a method to perform this database conversion process automatically so that a domain expert can do it without requiring a programmer to help. A lot of computation can be saved, and more interesting questions can be answered, if the database is structured with respect to the types of questions the users ask.

7 Future Research

This is the first large scale domain-independent spatial query management expert system that we have built. There are many research topics to be examined as we continue to develop our system to meet the users' needs. We mention some major areas for improvements or further investigation.

- As mentioned earlier in the paper, our nearest neighbor search algorithm requires empirical testing, and, as it is a naive approach to the problem, a faster yet more complicated solution may need to be implemented.
- We would like to examine the search speeds that result for a variety of tree structures. Currently, our trees contain about 5,000 observations. We would like to run tests on trees that contain all the observations in the IUE Minilog (about 70,000 records).
- To show the domain-independent nature of our approach, we will be constructing rule bases for other catalogs from a variety of different satellite observations.
- It would be useful if a library of successfully handled queries were maintained so that if a rule base is altered, the previously correctly handled queries can be rerun to ensure that a fix does not introduce a new error where there once was no problem. This utility exists in various expert system building tools, such as EMYCIN [5,18].
- We will likely be moving our interface from the Suntools environment to the more portable X-Windows system.

8 Summary and Conclusions

The spatial data management expert system is a large scale domain-independent system that serves as an intelligent front-end to databases containing spatial data. There are two major components to the system:

- The first is a spatial search module which uses the spatial component of the data in the database to produce a tree which contains the primary keys of all the records in the database. The spatial tree is indexed by the spatial part of a user's query, and a list is returned of all the primary keys that point to records meeting that spatial criteria. We find that the time required to process a user's query is reduced dramatically, compared with the time needed by a DBMS that must necessarily search its entire database to discover which records satisfy the user's spatial demands.

- The second is a domain-independent query expert system (QES) that uses a domain-specific rule base to preprocess the user query, effectively mapping a broad class of queries into a smaller set that is manageable by a commercial natural language processing product. QES uses a forward-chaining inference engine that relies on the specificity of the rules and an indexing scheme to achieve a quiescent state rapidly, thus transforming the user's English query into a form that can be handled by DataTalker.

This system is a step toward automatically building intelligent user interfaces for the large, non-homogeneous databases that exist today and are being planned for the future. We feel we have shown that the techniques we have used can be incorporated into working application systems immediately. Systems which force users to use specialized database query languages to access any data should be rethought.

Acknowledgements

We would like to thank Dr. Michael Van Steenberg (National Research Council) for helping us with the IUE Minilog, providing us with sample user queries, and assuring us of the existence of a user base that needs this type of system. We are indebted to Craig Goettsche, of Science Applications Research (SAR), for tackling Suntools and implementing the region selection module and a help module. David Kortenkamp, currently a graduate student at the University of Michigan, helped design and implement some of the spatial search module during his summer internship with SAR. Finally, we would like to thank William Campbell (NASA/GSFC), Scott Wattawa (NASA/GSFC), Scott Hill (SAR), Nicholas Short, Jr. (NASA/GSFC) and Larry Roelofs (Computer Technology Associates) for their discussions and inputs into this research.

References

[1] Bentley, J. L., Multidimensional binary search trees used for associative searching, *Communications of the ACM,* **18** 9, 509-517, September 1975.

[2] Bentley, J. L., Programming pearls, *Communications of the ACM,* **28** 11, 1121-1127, 1985.

[3] Brownston, L., E. Kant, R. Farrell and N. Martin, *Programming Expert Systems in OPS5*. Reading, Massachusetts: Addison-Wesley, 1985.

[4] Campbell, W., N. Short, Jr., L. Roelofs and S. Wattawa, The intelligent user interface for NASA's advanced information management systems, *Third Conference on Artificial Intelligence for Space Applications, Part II*, 1987.

[5] Davis, R. and D. B. Lenat, *Knowledge-based Systems in Artificial Intelligence*, New York: McGraw Hill, 1982.

[6] Finkel, R. A., and Bentley, J. L. , Quad trees: a data structure for retrieval on composite keys, *Acta Informatica*, **4**, 1-9, 1974.

[7] Floyd, R. W. and Rivest, R. L., Expected time bounds for selection, *Communications of the ACM*, **18** 3, 165-172, 1975.

[8] Friedman, J. H., Bentley, J. L., and Finkel, R. A., An algorithm for finding best matches in logarithmic time, *Stanford CS Report*, 75-482.

[9] Hendrix, G. G., E. D. Sacerdoti, D. Sagalowicz and J. Slocum, Developing a natural language interface to complex data, *ACM Trans. on Database Systems*, **3** 2, 105-147, 1978.

[10] Knuth, D. E., *The Art of Computer Programming, Vol. 3: Sorting and Searching*, Addison-Wesley Publishing Company, 1973.

[11] Martin, P., D. Appelt and F. Pereira, Transportability and generality in a natural-language interface system, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, 573-581, Los Altos: William Kaufmann, Inc., 1983.

[12] McDermott, J. and C. Forgy, Production system conflict resolution strategies, in D. A. Waterman and F. Hayes-Roth (eds.), *Pattern-Directed Inference Systems*, New York: Academic Press, 1978.

[13] Natural Language Incorporated, Natural language database retrieval system, version 3.0, Natural Language Incorporated, Berkeley, CA, 1988.

[14] Overmars, Mark H., and van Leeuwen, J., Dynamic multi-dimensional data structures based on quad- and k-d trees, *Acta Informatica*, **17**, 267-285, 1982.

[15] Short, N. Jr. and S. L. Wattawa, The second generation intelligent user interface for the crustal dynamics data information system, Telematics and Informatics, **5** 3, 253-268, 1988.

[16] Stroustrup, B., *The C++ Programming Language*, Addison-Wesley Publishing Company, 1986.

[17] Sybase Inc., Sybase database management, Sybase, Inc., Berkeley, CA, 1987.

[18] van Melle, W., A domain-independent system that aids in constructing knowledge-based consultation programs, Ph.D. dissertation, Stanford University, 1980.

BLANK PAGE

NATURAL LANGUAGE PROCESSING AND ADVANCED INFORMATION MANAGEMENT

James E. Hoard

Boeing Advanced Technology Center
P.O. Box 24346, MS 7L-64
Seattle, Washington 98124

1.0 Introduction. It is widely recognized that the development of sophisticated natural language interfaces (NLIs) will have great potential for users of complex information management systems. Such NLIs would largely alleviate the need to learn and use complicated access protocols. More importantly, such NLIs would also alleviate the need first to learn and then spontaneously to recall the names of hundreds, even thousands, of table and field names and their interconnections within a given information management system. Indeed, if one seriously entertains the notion of integrating several complex information systems into a larger whole, adding, say, application software packages for data analysis (numeric, symbolic, or both), the use of NLIs becomes almost mandatory. Note, too, that, in principle, there is no reason that the information management systems themselves need be of the same type. One can easily envision the need and desire to integrate the information in relational (and other kinds of structured) databases with the information in such diverse sources as CAD/CAM systems and text documents.

Granted that the user of an integrated system can expect to issue requests for action through an NLI, a fair question is this: What is the glue that can bind everything together behind the interface, the means by which just the right information is extracted from among a number of different information sources, put into the correct form and given to the appropriate application software packages, and then correctly processed by the application software? I advance the thesis that, in the long run, the glue must be natural language, that there is no other means for us to describe and understand the world and what goes on in it, and that there is no alternative means by which computers can be expected to do so. To be sure, one can write special procedures that directly integrate selected information sources with selected application packages (bypassing understanding, as it were), but I maintain that all such integrations must forever be ad hoc endeavors which are neither general, flexible, nor extensible.

Section 2 discusses the concept of Advanced Information Management (AIM)---the attributes it must have and the capabilities it will require. Section 3 presents the basic requirements for an adequate natural language processing (NLP) system and discusses some of the work we are doing at the Boeing Advanced Technology Center towards achieving robust natural language understanding. Section 4 presents a very high-level AIM architecture. Section 5 offers a brief summary and some concluding thoughts on how to go about developing a simple, but general, AIM system.

2.0 Advanced Information Management. It is commonplace to note that software application packages typically produce and process data in a particular format and, moreover, that they run only on a particular kind of computer using a particular operating system. Integrating, in a principled way, application software packages that were not originally designed to work together will require advanced information management. By advanced information management I mean the ability:

1) to query (or access) diverse information sources through a multi-modal, integrated user interface
2) to have the system access information automatically across heterogeneous computers, operating systems, and networking systems
3) to have the system perform automatically certain numeric and symbolic calculations, implied by the query, over the accessed information
4) to have the system present automatically to the user the results of the calculations in a meaningful and serviceable way.

2.1 Accessing Diverse Information Sources. Information sources, which either are or can be put in electronic form, range over a wide spectrum of types: text documents, drawings, schematics, photographs, movies, printed and recorded music, flat-file databases, hierarchical databases, relational databases, semantic-net knowledge bases, and so forth.

Clearly, fashioning an electronic card catalog that offers unique addresses and descriptions for (potentially) billions of information sources, complete with all relevant cross-references, is not a trivial task. Nonetheless, the complex addressing scheme that Theodor Nelson and his associates have developed to access the universe of information sources--Nelson refers to this as the "docuverse"--seems to be fully adequate to the task [1]. In Nelson's scheme, addresses are divided into four sections: Server, User, Document, and Contents. The Server section indicates where the material is located, physically or logically. The User section indicates the owner and other control and security information. The Document section provides the logical entity (i.e. name, version, etc.) under which the information is stored. The Contents section describes the material and includes in hypertext fashion all the links to directly related material.

The docuverse as a whole will not be catalogued and made accessible anytime soon. Indeed, issues of privacy, public safety, and national security suggest that access to everything is, in any event, undesirable. Within a number of spheres, however, a well- and consistently-catalogued docuverse is highly desirable. Examples among docuverses that could be publicly available include those for various academic disciplines, unclassified government scientific programs, and court proceedings. Proprietary docuverses will include those of many government agencies and of virtually all individual manufacturing and service industries.

For an AIM system to make use of the items in any particular docuverse, a simpleminded Contents section will not do. In order for an AIM system to access and make use of a docuverse of diverse information types, the Contents section of the address scheme must contain links to both a Structural Description (SD) of the material and a Real-world Interpretation (RWI) of the structural description. To see why we need both of these, consider as a document some instance of a relational database. (At this level of granularity, we can consider the set of files to be a hypertext document.) Then the SD indicates 1) that the document was created by using a particular database system, say, ORACLE, and 2) which files are data files and which files give the complete database definition of this particular instance of ORACLE. Now, in and of itself, a database definition has no intrinsic meaning, since, in general, table and field labels are not self-explanatory. Although mnemonic labels obviously help those familiar with a particular database to remember what it contains, table and field labels could as well be arbitrarily chosen numbers (say, tables 1 through 157 and fields 1 through 905) for all the help they afford the integration task. Only by providing a complete RWI for a structural description can database definitions be interpreted by machines (and, by people) unfamiliar with the design and contents.

It is easy to see that having a RWI of the information in a database is crucial to integrating the information with information from other sources. For instance, given a field label such as "DATE", it is natural to ask what it is that "DATE" is the date of and to inquire about the interpretation of the internal structure, if any, of the values in the "DATE" field. Compare "760704", "040776", "04-07-76", "070476", "07-04-76", "04/07/76", and "07/04/76" as possible ways to express "July 4, 1776".

There are many other document types beside databases that require explicit RWIs. Among them are CAD/CAM documents. Given that a document is a drawing created using some CAD/CAM system, say, CATIA, then the SD indicates that it is indeed a CATIA drawing and gives descriptions of its files. The RWI that parallels the SD provides the basis for integrating the information in the drawing with the information in other documents of other types.

The RWIs of databases, CAD/CAM documents, and many other materials in a docuverse will have to be given in some natural language (English is the obvious choice for English speakers) for one simple reason: Natural language is is the only universal language people have, the only kind of language there is for describing anything at all, whether it be a description of some state of affairs, real or imagined, or of some action we wish to take. This means that natural language offers the only way we can realistically hope to integrate a large number of distinct information sources in an automated fashion.

To see what an RWI looks like, consider a database of geographical information. Suppose the SD reveals that we have a table, labeled GEOGRAPHICAL-DATA, which contains, among others, fields labeled PLACE and ALTITUDE. The RWI of these labels might be as follows:

> ALTITUDE is the altitude of PLACE in feet above mean sea level; PLACE is the name of a geographic feature (mountain, lake, city, airport, ...); GEOGRAPHICAL-DATA contains the names and geophysical attributes (see ALTITUDE, LOCATION, AREA, ...) of a number of geographic features (see PLACE) of North America.

If we have a natural language understanding system that can interpret the RWI and put it into machine-usable form, then we have, in principle, a way of integrating information from this information source with that of any other.

Unlike databases and CAD/CAM systems, text documents contain, by and large, their own RWIs by virtue of being text documents. I say "by and large", since text information that is presented in tables, indexes, tables of content, chapter and section headings, headers, and the like will require RWIs unless they conform to default formats known to the AIM system. In addition, texts often contain pictorial and graphic insertions that are not, in general, self-explanatory and, for which, RWIs will be required.

In sum, using Nelson's docuverse addressing scheme, within a given docuverse, an AIM system will automatically assign document addresses to all new items that are created and will automatically update the addresses as documents are revised. There is, however, one crucial point: It will be the responsibility of the creators and revisers of documents to provide and maintain the integrity of the RWIs in the Contents section of the addresses, since there is no way that any AIM system can read minds.

2.2 Executable Documents. Many of the items in the docuverse are not static, run-of-the-mill materials, i.e. unformatted text, graphics, database files, or whatever. They are, in fact, executable programs, materials that from a docuverse perspective can be viewed as Executable Documents (EDs). Such programs run the gamut from the simplest COBOL or C program to massive expert systems and FORTRAN programs. Since the docuverse address scheme allows us to link documents at will, we can link together compiled code, source code, and descriptive material in hypertext fashion. Now, if, in addition, we can prepare and link to an executable document an Input-Output Document (IOD), a document specifying a program's input and output requirements and behavior, and an RWI describing the IOD, we can entertain the notion of integrating data and programs that were not originally designed to work together. Moreover, we can hope to do so with methods that are not ad hoc and do not rely on brute force.

In particular, we could avoid brute force, "one-off", non-general procedures that map words like "high" and "height" directly onto a database field like ALTITUDE. Rather, we would invoke general natural language processing procedures to relate the word "altitude", taken from the RWI of ALTITUDE, to vertical elevation and height. Since the RWI also tells us that the altitude is expressed in feet, an AIM system can also report the units along with the value. In fact, if we have another database, say, a geographic database for Europe, and we have its SD and RWI, we can use our general methods to find out the altitude of the Matterhorn. Suppose now that we wish to know which mountain is higher, and by how much. If the value for the height of the Matterhorn is expressed in meters--and an AIM system will know that it is from the RWI--the AIM system will invoke an ED for converting units, do the conversion, compare the altitudes, and report the results in feet or meters, as we desire. (Not having an AIM system with geographic databases in its docuverse, I had to do all the work myself. As it happens, the Matterhorn is about 112 meters, or 369 feet, higher than Mt. Rainier.)

2.3 Heterogeneous Computer Systems. Obviously, to access very much at all of the docuverse, an AIM system will immediately be confronted with the fact that materials tend to reside on different computers with different operating systems. To make matters worse, the threads that stitch computers together come in different weights and colors, e.g. the Network File System (NFS) and the Transmission Control Protocol/Internet Protocol (TCP/IP). While a successful AIM system will require a number of IODs and RWIs in order to move information round and about, I will not pursue the matter here.

2.4 Implied Calculations. Every interesting use of an AIM system requires that the system have the ability to do implied calculations. Some of the queries one might direct at an AIM system will be of a simple sort. For example, one might inquire: "What is the height of Mt. Rainier?" Given access to a geographic database, we surmise that the system will simply look up the answer in a table of mountain names and heights and report the value to us, viz. "14,411 feet". Suppose the query were framed with less specificity, say, "What is the tallest mountain in Washington, and how tall is it?". On the assumption that the database tables contain information correlating mountain and state names, the system is now required to perform some very simple implied calculations, both numeric (comparing heights of mountains) and symbolic (distinguishing mountains in Washington from those in other locales).

In principle, queries containing implied numeric and symbolic calculations can be as elaborate and as oblique as we like. For example, it would be quite reasonable for a geologist to ask: "How much did the North American Plate move with respect to the Pacific Plate in 1988?" [2]. The first task for the AIM system is to understand the query. This requires symbolic processing of a high order and is the topic of section 3. Suppose for the sake of argument that the system succeeds in understanding the query. Then, if data files are available for 1988 that give range values from satellites to ground stations on each plate as well as the angles between the ground stations and the satellites, fetching the data and performing some numeric calculations involving the sine of this and/or the cosine of that will yield the answer the geologist seeks.

It should not be supposed that the worth of natural language symbolic processing is confined to natural language interfaces--or even, as I have asserted, to gluing together diverse computer packages. On the contrary, the vast majority of the world's information sources are text documents, as a trip to any library, perhaps even to one's study, will easily reveal. A real AIM system will answer queries that require symbolic calculations over the text data that is included in the docuverse. For example, suppose we have a biography of Abraham Lincoln in electronic form. If so, it should be quite reasonable to ask such questions as: "Did Lincoln ever visit Richmond, Virginia?", "How many terms did Lincoln serve in Congress?", and "Who killed Abraham Lincoln?". In general, the answers will not be found by simplistic pattern matching techniques. It would probably be difficult indeed to find a biography of Lincoln that contained the sentence "John Wilkes Booth killed Abraham Lincoln," or even one that stated "It was John Wilkes Booth who killed Abraham Lincoln." Instead, we have all, I expect, read the narrative accounts of John Wilkes Booth sneaking up a back stairway in Ford's Theater, entering Lincoln's box, shooting Lincoln, leaping to the stage and breaking his leg, making his escape, and of Lincoln being carried across the street and dying early the next morning. In short, answering even such apparently simple queries such as "Who killed Abraham Lincoln" will require sophisticated natural language processing that can comprehend narratives (understand each of the sentences and the interconnections among them) and is able to use semantic and real-world knowledge to draw conclusions (e.g. "X kill Y" means "X cause Y to die"; "If X does Z, and Z causes Y to die, then X kill Y") .

Clearly, there will be limits on the implied calculations that even a mature AIM system could be expected to perform automatically. Basically, we expect such a system to be able to do what an able human assistant could do. Thus, it would obviously be futile to ask: "Is Fermat's last theorem true?" or "Will the universe ultimately collapse?". At best, the system would respond with "No one knows.", or something of the sort. On the other hand, there is a broad spectrum of relatively mundane tasks that an AIM system ought readily to perform. Among these are reading vast numbers of text documents for content and applying a variety of EDs to data gathered from many different information sources.

2.5 The Interface. The interface of an AIM system will need to be very capable. Issues involving user modeling, avoiding cognitive overload, and the appropriateness of input and presentation techniques are beyond the scope of this paper. Nevertheless, a few remarks on input and output are required.

First, a sophisticated natural language interface is a necessary part of any AIM system. While I have nothing against icons and desktop metaphors, there are obvious limitations to iconic interfaces--viz., they are virtually useless for formulating syntactically complex commands. While command languages do not suffer from that limitation, they suffer from two other undesirable attributes: 1) they tend to be hard to learn and use, and, more fundamentally, 2) each of them is artificial and, hence, has no intrinsic semantic interpretation. For example, on a SUN workstation, you can invoke a command line to put a representation of a clock on the screen at the location of your choice. The command line is best described as arcane and can be compared to an AIM system command, which we might frame as "Put a clock in the upper right-hand corner of the screen". In a multi-modal interface we could frame the request as "Put a clock there", with an appropriate gesture to indicate where "there" is.

Second, the presentation component of an AIM system must evaluate the characteristics of the information that is accessed, presenting it to the user in a comprehensible and appropriate manner. The characteristics of interest include the number of elements and whether they are numbers, text, or graphics. A smart presentation manager will be able to make many correct choices on its own and adjudicate the form of the presentation with the user, suggesting alternatives on a case by case basis, as required. Suppose, for instance, that, not realizing the extent of the work done on the Athapaskan languages, I asked my AIM system for a listing of all the linguistic work done on this family of languages. The presentation manager should inform me that it has found many hundreds of items to include in an Athapaskan linguistics bibliography and give me the opportunity 1) to narrow my query or 2) to group the items in some convenient way and put them into a file for browsing. Similarly, an AIM system should make sensible choices in deciding how a set of data values is best presented graphically, deciding from the nature of the data to use, say, a bar chart, histogram, or scattergram.

2.6 AIM Requirements. The requirements of an AIM system can be summarized as follows: We need a universal addressing scheme, such as the one Nelson has devised, to keep track of the things in a docuverse. We need an adequate NLP system, and we need it both for the interface and for interpreting what static documents contain and what EDs can do.

We will need what appears to the user to be an "intelligent agent" that can use natural language descriptions and instructions to carry out our wishes. Figure 1 shows the overall AIM system. Note that as far as an AIM system is concerned, there are only three kinds of things in the docuverse: stand-alone EDs, e.g., application programs like statistics packages and grammar and style checkers; stand-alone static documents, e.g. text documents; and application packages that consist of static documents and the associated EDs that are used to manipulate the data in the static documents. To make use of an application package, an AIM system will query the RWIs and SDs of the static documents and the associated ED as information sources, then use the application package ED to access the data in the static documents.

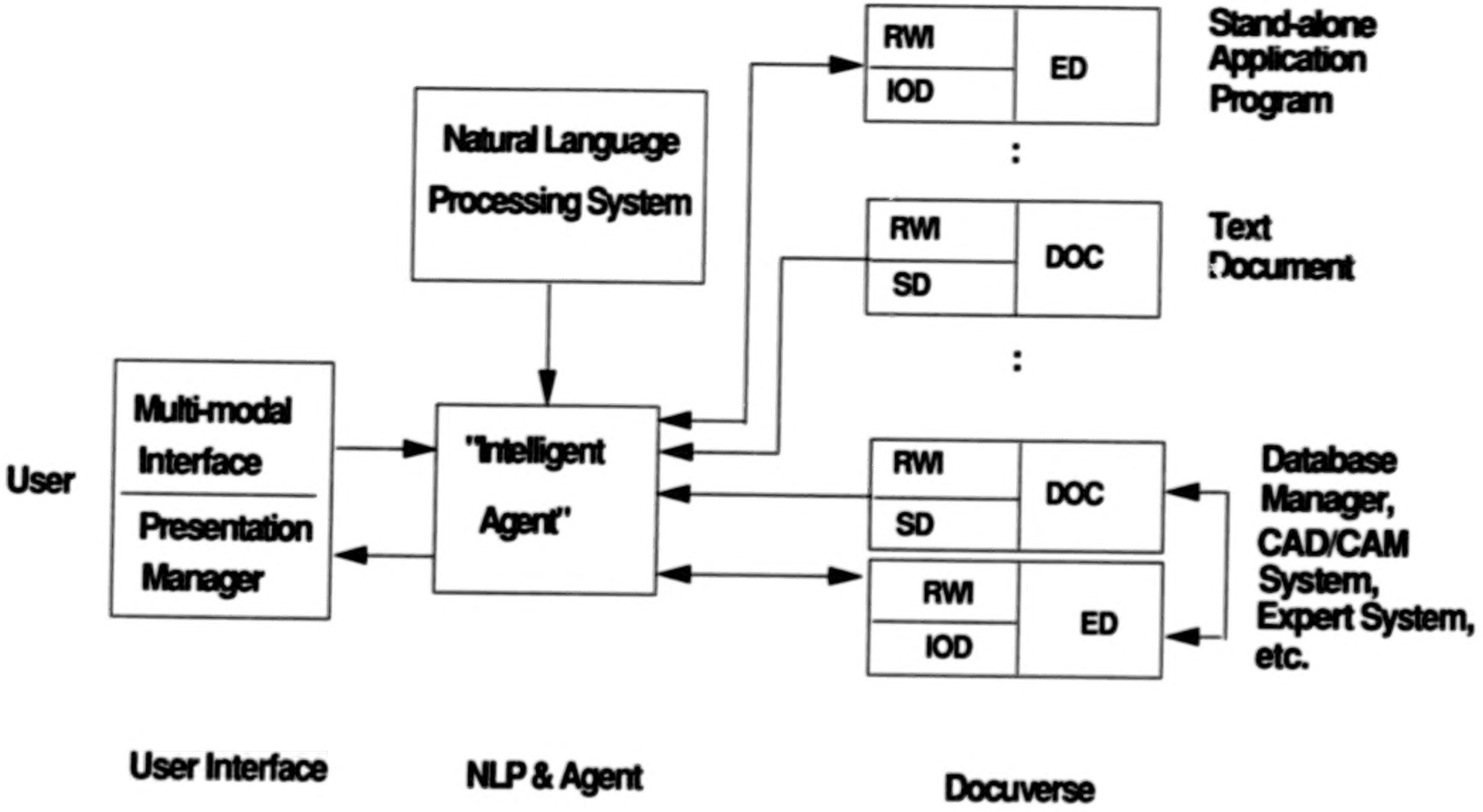
3.0 An Adequate Natural Language Processing System. In our work on natural language processing at Boeing, we base our approach on three fundamental premises: 1) all linguistic forms have meaning--there are no meaningless linguistic elements; 2) linguistic and real world knowledge can be described in discrete syntax, semantics, and pragmatics modules; and 3) natural language processing must interleave linguistic and real world knowledge.

The first premise is easy to justify. Consider the pitfalls in ignoring the difference between infinitive and gerund constructions revealed by such pairs as: "They stopped to search for survivors" and They stopped searching for survivors". In the first example, the infinitive gives the reason for stopping; i.e., semantically, "to search for survivors" is a 'purposive' modifier of the intransitive verb "stop". In the second example, the gerund phrase "searching for survivors" is the direct object of the transitive verb "stop"; semantically, it is the 'range' complement of "stop", a complement that states the activity that was stopped. Similarly, consider the pair of examples "Reagan sent a message to Iran" and "Reagan sent Iran a message". For the first sentence of this pair, Reagan's message could have been sent anywhere in Iran, although it could have been sent to the government of Iran. The second sentence states unambiguously that the message was sent to the government of Iran. This pair of sentences underscores the need to take account of subtle meaning differences occasioned by seemingly innocuous differences in word order and the presence or absence of words like "to".

The second premise stems from the observation that whatever the subject matter, the English language is, minor dialect differences aside, everywhere the same and that the work of developing general syntax, semantics, and pragmatics modules need be done only once. When such a system is extended to a new domain, only the knowledge specific to that domain need be added to extend the system's coverage. The conceptual distinction between syntax, semantics, and pragmatics embodied in the system allows us to develop each module separately and incrementally--we can make progress in one module while we resolve problems in another. Moreover, we can keep declarative knowledge, say, the rules of English syntax, quite separate from the procedural knowledge required for an efficient syntactic parsing algorithm. Thus, we can expand and revise the syntax rules without affecting the parser, and conversely, we can improve the parsing algorithm without affecting the syntax rules. Indeed, extending our syntactic rules to cover telegraphic speech (e.g."Having wonderful time. Wish you were here.") was accomplished by adding a few rules to the grammar for ordinary text, without making any change to the existing grammar or to the parser.

Our third premise is an explicit acknowledgment of the inherent massive ambiguity of natural language, ambiguity that is at once its strength--for it allows us to say whatever we wish with decidedly finite means--and its weakness--for it is all to easy to misinterpret what one hears or reads and to be misinterpreted in turn. Out of context, even relatively simple sentences can be very ambiguous. "The girl saw the boy on the hill with a telescope" is a typical example. Either the boy or the girl or both is on the hill, and either the boy or the girl or the hill has a telescope, which the girl may or may not have used to see the boy. It seems to me that only by interleaving the application of linguistic and real-world knowledge word by word and phrase by phrase during text (or speech) processing can we hope to reduce to tractability the amount of computation required for disambiguation. In any event, a system that interleaves syntactic, semantic, and pragmatic knowledge, making disambiguation decisions as soon as possible in every instance, can converge on the best interpretation of a text in some given context much more quickly than can any system that mechanically constructs all possible syntactic parses, then constructs all the possible semantic interpretations of all the possible parses, and then decides upon the best text interpretation from among all the possible interpretations.

3.1 The Sapir Natural Language Processing System. In our efforts to develop the Sapir natural language processing system (named in honor of Edward Sapir, the eminent linguist of the first half of the 20th century) we have endeavored to utilize the best current knowledge about the properties of natural language in general and of English in particular. As shown in Figure 2, we are implementing our linguistics-based approach in four principal modules: token and layout analyzer, syntactic analyzer, semantic analyzer, and pragmatic analyzer. The components of the system are designed to interact cooperatively through the message workspace.



Stand-alone Application Program
RWI
IOD
ED
Natural Language Processing System
Text Document
RWI
SD
DOC
User
Multi-modal Interface
Presentation Manager
"Intelligent Agent"
Database Manager, CAD/CAM System, Expert System, etc.
RWI
SD
DOC
RWI
IOD
ED
User Interface
NLP & Agent
Docuverse

Figure 1. Overall AIM System

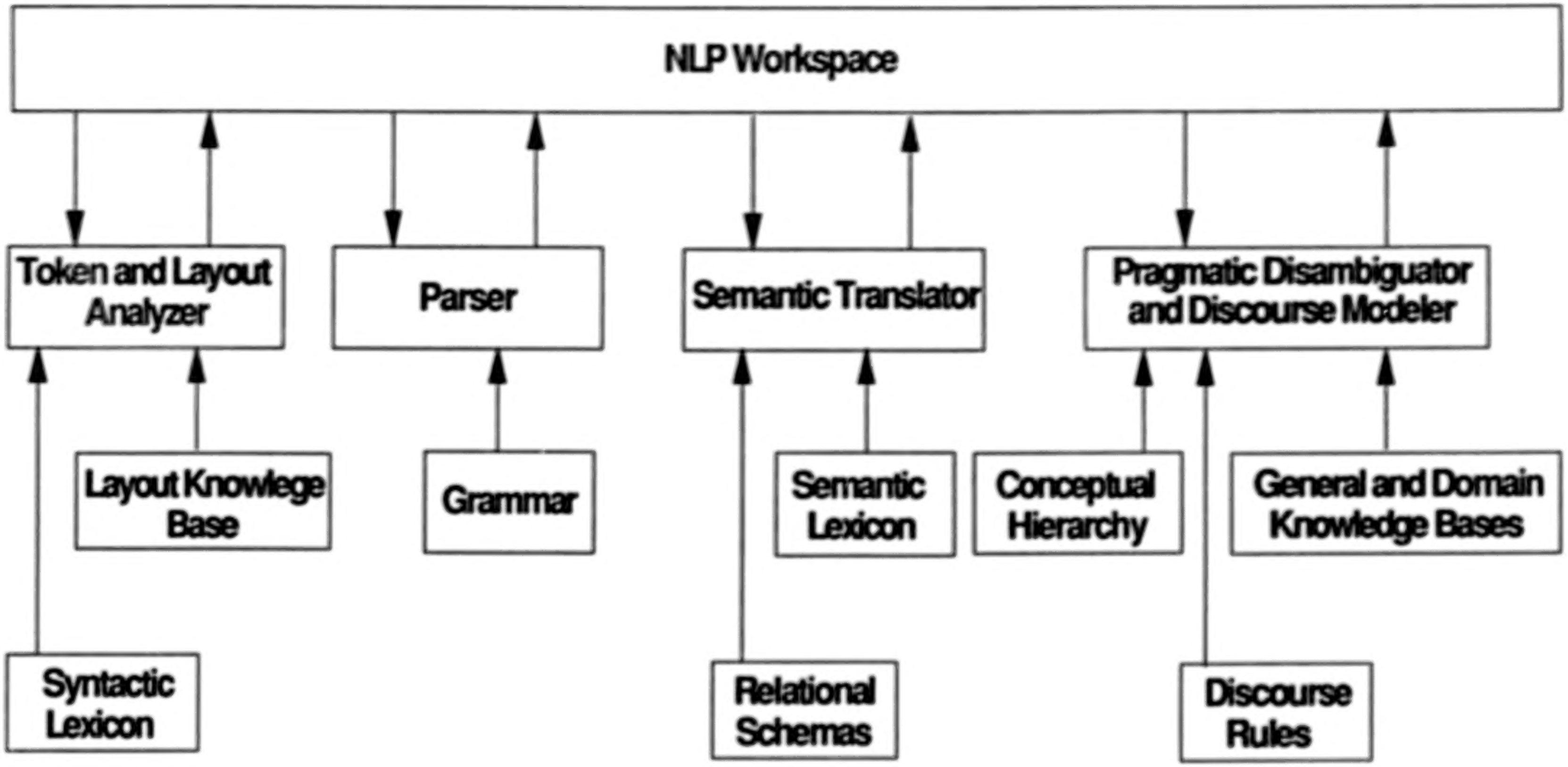

NLP Workspace
Token and Layout Analyzer
Parser
Semantic Translator
Pragmatic Disambiguator and Discourse Modeler
Layout Knowlege Base
Grammar
Semantic Lexicon
Conceptual Hierarchy
General and Domain Knowledge Bases
Syntactic Lexicon
Relational Schemas
Discourse Rules

Figure 2. Sapir System Architecture

The token and layout analyzer initiates the text analysis process by recognizing each word in the input string in turn and posting a word packet (the word, its position, and its syntactic characterizations) in the message workspace. The parser and grammar, working bottom-up and left-to-right, use the word packets to produce syntactic constituents, each of which is posted into the message workspace as soon as it is constructed. The semantics module takes each of the syntactic constituents in the order in which they appear, provides all possible possible semantic translations, and posts the semantic structure in the message workspace, linking each interpretation with its syntactic counterpart. The pragmatics module takes each syntactic-semantic structure in turn and rates it for pragmatic likelihood. Those syntactic-semantic structures that are deemed acceptable are used for further processing by the syntax and semantics modules; those deemed unacceptable are held aside and, hence, are not carried forward as potential sub-constituents of larger syntactic- semantic structures. Viable syntactic-semantic structures posted in the message workspace are available to the discourse modeling component to analyze as potential discourse constituents. Thus, our system architecture deals forthrightly with the inherent massive ambiguity of natural language, eliminating unlikely interpretations of a text contextually as soon as possible during the analysis process and reducing to a minimum the number of structures that must be carried downstream, at great computational expense, during the course of the text understanding process.

We are currently evaluating our interleaved architecture in a blackboard environment, using Boeing's Erasmus Blackboard, which is configurable in a variety of ways [3]. The advantages of a blackboard implementation are fourfold: 1) we can run the modules in distributed fashion across several computers; 2) the system modules can be implemented in different languages; 3) we can experiment with interleaving (and parallelizing) the computations; and 4) we can experiment with opportunistic problem solving techniques which would allow our system to dynamically decide where to focus its attention.

3.1.1 The Token and Layout Analyzer. The token and layout module processes ASCII files and infers typographical, structural, and lexical information from the printed form of documents. The current output of the layout analysis subsystem of this module is a demarcation of sentence boundaries. When it is mature, the layout analyzer will recognize and characterize text constituents such as sentences, paragraphs, and sections, and will model information about text structure in concert with the discourse analyzer. The token analysis subsystem of this module uses finite-state recognizers to identify character strings as tokens and converts the tokens into lexical items. Ambiguities are passed up to a token parser that resolves them on the basis of the larger textual context. The token parser can postulate both single and multi-word lexical items from token strings, and, conversely, it can map the substrings of a single token into individual lexical items. At present, we have a lexicon with syntactic information on over 11,000 words. Even though the number of words in our lexicon is increasing rapidly, the lexicon will never be complete, since previously unencountered proper names will always appear in new text material, and new words, especially single- and multi-word proper names, are, in any event, constantly being added to the language. Thus, at the user's discretion our present token analyzer can default an unrecognized token to the category proper noun or ask the user to identify it interactively.

3.1.2 The Syntax Module. The syntax module provides a domain- independent description of English syntax. Our approach is based on Generalized Phrase Structure Grammar (GPSG) [4], a high-level formalism in which each grammar rule is equivalent to a large number of traditional context-free phrase structure rules. Without this formalism, it would be necessary to write tens of thousands of phrase structure rules. GPSG is especially well suited to natural language processing since it allows parsing to proceed in a context-free, bottom-up manner. Our syntax module, which provides very broad coverage of English, consists of approximately 300 grammar rules and an efficient parser [5] that we believe is one of the fastest in existence. We have developed special parsing methods that make it unnecessary to convert the high-level GPSG-like grammar into its expanded form, making it possible to develop the grammar interactively, an indispensable aid for iteration, testing, and validation of possible rules.

3.1.3 The Semantics Module. The semantics module provides semantic translations of syntactic constituents. We have designed and built a prototype semantic translation program which takes in turn each syntactic constituent constructed by the parser and produces a semantic interpretation of a sentence. The module generates semantic structures by pairing semantic translation functions associated with each lexical element with every syntactic configuration in which the lexical item can be a leaf node. The initial semantic translation of a sentence is then the semantic structure built compositionally from the individual syntactic configurations and lexical elements that are its constituents. The semantic representations are expressed in a formalism, called relational logic, that we are developing for natural language semantics. The formalism uses a special vocabulary of relational operators that link lexical items into semantic structures. Relational logic structures make explicit both modifier-head and predicate-complement bindings and can be decomposed algorithmically to derive both lexical and predicational inferences. Lexical inferences are made by consulting basic definitions, expressed as relational logic structures, and invoking general substitution procedures.

The relational logic representation we have developed offers a holistic view of semantic structures, explicitly giving the cognitively salient relation (or function) that obtains between the linked elements of the semantic structures. While the cognitive view of language it reflects has its origins in antiquity and traditional grammar, the recent precursors of relational logic include writings as diverse as Fillmore [6], Chafe [7], Quirk et al. [8], and Hudson [9].

Relational logic is very much in the spirit of what is now termed cognitive linguistics. The theoretical underpinnings of cognitive linguistics have been carefully set forth by Langacker [10]. The essential notions are 1) the distinction between autonomous and dependent elements and 2) the asymmetry between these two kinds of elements (called A/D asymmetry). Autonomous linguistic elements are those that require no elaboration to be semantically complete. The most prototypical autonomous elements are the nouns. Dependent elements are those that cannot stand alone, that obligatorily require elaboration. The prototypical dependent elements include verbs and auxiliary verbs, articles, adjectives, adverbs, and prepositions. The categorizations of linguistic elements as autonomous and dependent is not rigid. A particular linguistic element can figure autonomously in one grammatical construction and dependently in another. A grammatical construction typically has at least one element that functions dependently and one that functions autonomously; i.e., A/D asymmetry is an observed property of the vast majority of grammatical constructions. Langacker also gives adequate characterizations of the notions modifier, head, predicate, and complement--characterizations that are indispensable for adequate and accurate semantic analysis, and which we have adopted in our system of relational logic.

Figure 3 shows the relational logic semantic graph for the sentence, "How much did the North American Plate move with respect to the Pacific Plate in 1988". The query is one that a geologist might ask of NASA's Crustal Dynamics Data Information System, as described in [2]. The nodes of the graph are the linguistic elements and the arc labels are the relations. Dependent elements are placed above autonomous elements: thus modifiers are shown above the heads they modify, and complements are shown below their corresponding predicates. The relations are drawn from a small, finite set and serve to indicate the overall cognitive (or conceptual) meaning of the sentence.

The semantic ambiguity problem for a natural language processing system is twofold--first, to have a means of representing functional ambiguity, and second, to provide a way of choosing from among the semantic alternatives. Relational logic provides a quite natural way of representing functional ambiguity. For example, the prepositions "of", "by", and "for" have a number of the relational meanings, as shown in Figure 4. We view the semantic relation between the preposition and its first argument as holding between the element that typically precedes the preposition in syntactic configurations and the entire nominal phrase that follows the preposition. Thus, in "a test of skill" the "test" is "with respect to" "skill"; and, in the "most of today", "most" is the "extent of" "today". While "of" seems to have little inherent lexical content, the prepositions "by" and "for" have intrinsic meaning, i.e. they have different senses, in each of their uses. For instance, in a locative use, "by", as in "by the window", means roughly "near", and "for" in its directional use, as in "leave for London" means roughly "towards". In a temporal use, "by", as in "by Tuesday", means roughly "not later than", and durational "for", as in "sleep for hours" means roughly "during a time interval of". The various senses of a word are also expressed by relational logic structures. This provides a basis for deriving natural language inferences among words and sentences, as discussed in [11].

Relational logic also provides an efficient means for choosing among alternative semantic structures. Consider the sentence, "John thought for days of the incident by the river." The main verb "think" enters into complement structures such as 'think(cog:X)' and 'think(cog:X, r:Y)', where 'cog' means "cognizer" and 'r' means "range". Since "think" is used intransitively in this example, our system selects the first of these complement structures, 'think(cog:X)'. The possible modifiers of "think" include both outer situational locatives and temporals (those that specify a location in space or time, respectively) and such inner circumstantial modifiers as 'wrt' ("with respect to"), 'man' ("manner"), and 'ext' ("extent"). (Here we follow Barwise and Perry [12] and Pollard and Sag [13] in distinguishing situations from circumstances.) Our procedures must choose among the alternative possible prepositional phrase interpretations based partly on the range complements of the prepositions and partly on the modifiers permitted by the main verb. In analyzing the prepositional phrase "for days", the range of "for", namely "days", makes it likely that the durational interpretation of "for" is intended and that the other possible interpretations of "for" are unlikely in this context. We further interpret the phrase "for days" as an inner circumstantial modifier of "think", since it does not specify a point on the spatio- temporal plane (as would, say, "in 1988" or "on Tuesday"). Next, our program considers both attachment possibilities for the phrase, "of the incident": It must be associated with either the preceding noun to form the phrase, "days of the incident", or with the verb "think" to form "think of the incident". Our system recognizes "think" plus a 'wrt' modifier as a conventional pattern. Hence, the interpretation "thought of the incident" is chosen as the more likely reading. Similarly, the phrase "by the river" can be associated either with the preceding noun to form the nominal phrase "incident by the river" or with the

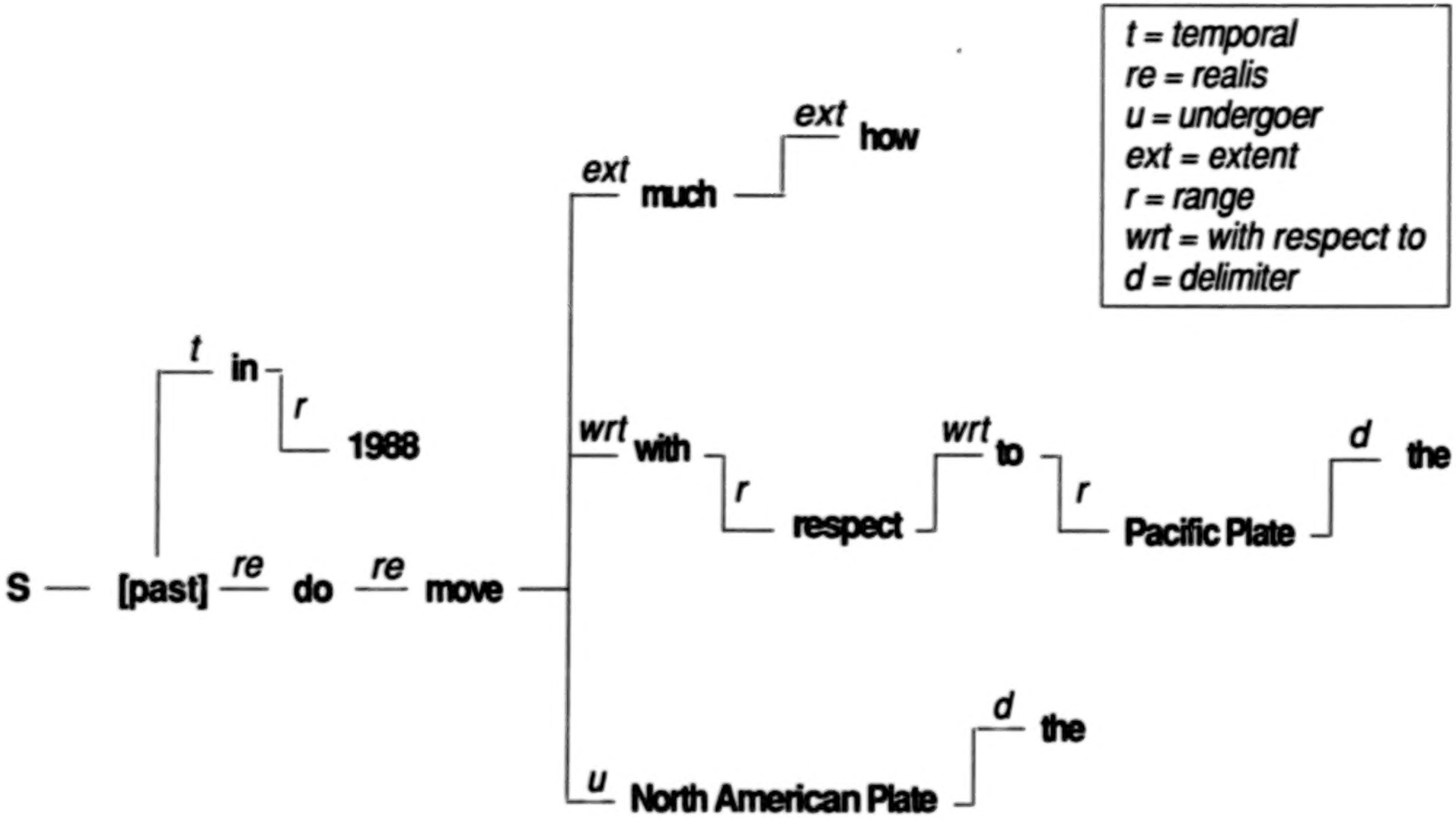

t = temporal
re = realis
u = undergoer
ext = extent
r = range
wrt = with respect to
d = delimiter
S
[past]
re
do
re
move
t
in
r
1988
ext
much
ext
how
wrt
with
r
respect
wrt
to
r
Pacific Plate
d
the
u
North American Plate
d
the

Figure 3. Semantic Structure of "How much did the North American Plate move with respect to the Pacific Plate in 1988?"

of

of (wrt : X, r : Y):	test of skill	of (wrt : test, r : skill)
	king of England	of (wrt : king, r : England)
	mayor of Chicago	of (wrt : mayor, r : Chicago)
	burning of coal	of (wrt : burning, r : coal)
of (att : X, r : Y):	man of courage	of (att : man, r : courage)
of (c: X, r: Y):	die of starvation	of (c : die, r : starvation)
of (ess: X, r: Y):	city of Chicago	of (ess : city, r : Chicago)
	month of August	of (ess : month, r : August)
of (ext: X, r: Y):	cup of water	of (ext : cup, r : water)
	most of today	of (ext : most, r : today)

LEGEND

ag = agent
att = attributive
c = causer
ess = essive
ext = extentive
l = locative
me = means
p = purposive
q = quantitative
sub = substitutive
t = temporal
wrt = with respect to

by

by (l : X, r : Y)	walk by the window
	over by the door
by (t : X, r : Y)	leave by two o'clock
by (wrt : X, r : Y)	call by name
	play by the rules
	go by plane
by (me : X, r : Y)	hit by a rock
	build (it) by hand
by (ag : X, r : Y)	kidnapped by terrorists
by (q : X, r : Y)	(divide) thirteen by two
by (ext : X, r : Y)	little by little

for

for (l : X, r : Y)	leave for London
for (t : X, r : Y)	sleep for hours
for (p : X, r : Y)	cake for Joan
	go for coffee
for (wrt : X, r : Y)	an eye for color
	big for his age
for (sub : X, r : Y)	stand (in line) for Mary

Figure 4. Functional Relations for Of, By, and For

main verb to form "thought by the river". The locative interpretation of "by" applies equally to both possibilities. Thus, the discourse model must be consulted to help make the decision: If it has knowledge of some event which occurred by a river, the nominal phrase would be sanctioned. Otherwise, the verbal attachment, as a situational modifier, will be preferred.

The foregoing example demonstrates our strategy for interpreting such constituents as prepositional phrases and for choosing among the potentially large number of alternatives. Our procedures first narrow down the possibilities to those that are most likely based on a functional analysis of the words. Then, reasoning and discourse considerations select the best choice from among the likely candidates. Relational logic serves as an effective representational formalism both for expressing functional ambiguity and as a basis for performing intrasentential disambiguation.

3.1.4 The Pragmatics Module. The pragmatics module has two principal functions. First, it provides the means to select (disambiguate) in context the most likely semantic structures from among those provided by the semantics module. Second, the pragmatics module builds and maintains a discourse model, interpreting the text as a coherent structure having causal, temporal, and spatial connections among its sentences. To do this, the pragmatics module must, among other tasks, establish different levels of event description in a sentence; establish discourse segments; resolve pronominal and definite references; and determine the most likely scope of quantifiers.

Our strategy for disambiguation initially uses a category hierarchy, matching categories against expectations to assess the likelihood of the hypothesized semantic relations among the lexical items in a given semantic structure. If several viable candidate structures remain, a reasoning system brings discourse and encyclopedic knowledge to bear to select the most likely meaning. For example, the transitive relational frames for "teach" are "teach(c:X, g:Y)" and "teach(c:X, r:Y)", where "c" is the "causer", "g" is the "goal", and "r" is the "range" complement of "teach". "John taught Bill" and "John taught algebra" illustrate the two possibilities. Ambiguous semantic translations arise, however, since it is not possible to tell a priori if the direct object of "teach" is the goal or range complement. An example is "John taught Wittgenstein", where we cannot be sure without considering the context whether John taught Wittgenstein's philosophy or Wittgenstein himself. In such cases we must appeal to the reasoner and the discourse model.

Our work on the pragmatics module is being undertaken jointly with Professor Lenhart Schubert of the University of Alberta and several of his students. The semantic net system that Schubert and de Haan have developed has a number of innovative features that enhance the efficiency of inference [14]. Without such innovative strategies, a natural language processing system would be overwhelmed by the large number of propositions in its knowledge base. The reasoning system uses an extended first-order logic that provides for event variables. The primary inferencer, called ECoLogic, is supplemented by a number of specialist reasoners [15] that reason about such things as anaphora and definite reference, quantifier scoping [16], temporal relations [17], color relations, and set enumeration.

The anaphora and definite reference program, for example, uses heuristics derived from Reinhart [18] and Bosch [19]. The program considers coreference possibilities among six different types of input phrases (or terms): pronouns, proper names, generic noun phrases, definite noun phrases, indefinite noun phrases, and quantified noun phrases. In addition, verbs are assigned "episodic constants" which can be the reference of pronouns and definite noun phrases (NPs). An example discourse that is handled by the program is: "John kissed Mary on the forehead. It embarrassed the young woman." The program assigns a numerical weight to each coreference possibility (a pairing of terms that can refer to the same object). It then lists the readings in order of preference, based on the combined weights derived from the heuristics. For the simple discourse above, the highest average weight is assigned to the reading in which the definite NP "the young woman" is coreferential with the definite NP "Mary", and the pronoun "it" is coreferential with the episode (the event of John's kissing Mary on the forehead) associated with the first sentence.

Our approach to discourse modeling uses as a starting point the observations on discourse structure offered by Reichman [20] and Webber [21]. A discourse is a text which coheres in a real-world context, semantically and pragmatically. The task of discourse modeling is to establish a coherent interpretation of a text. Our strategy is to develop disambiguation algorithms utilizing a number of specialist subsystems which place in context the relational logical structures generated by the semantics module. These subsystems will 1) decide what the discourse focus is at any given point in a text, tracking topic and focus shifts to establish discourse segments; 2) generate and maintain sets of potential referents both for entities mentioned explicitly in the text and for entities implicit in the text and derived by inferencing; and 3) discover and dynamically maintain a model of the discourse. We are concentrating our efforts initially on a few of the most common discourse devices, exploring ways of utilizing adverbial discourse cues (e.g. "for example", "thus", "first", and "on the other hand") as indicators of discourse structure and overt text format markers such as paragraphing, section numbers and titles, and the like.

4.0 An AIM Architecture. The obvious candidate for an "intelligent agent", as discussed in 2.6, is an Object-Oriented Database (OODB) or Object Server [22]. OODBs have the desirable property of being able to deal with documents, both static documents and EDs, as "objects". OODB objects are put into "classes", which define the kinds of operations, including any number of special "methods", that can be performed on them. To invoke any of an object's methods, one sends it a "message".

To integrate diverse documents in an AIM system requires a class of executable documents (or methods) that can be called Transform Documents (TDs). TDs use SDs and IODs to access data files (through an ED or directly), to export data to another ED, and to receive the results of computing done by an ED. In brief, a TD takes data in its original form, transforms it into the form expected by some ED, and directs the ED to use the data to compute some function or functions, which the AIM system knows the ED to be capable of because it has accessed and understood its RWI and IOD.

The TDs will be supplemented by Reasoning Specialists (RSs), in the manner of the NLP specialists, that analyze queries and docuverse RWIs to determine what actions to take in a given situation. Taking an anthropomorphic view of the matter, the RSs will do what a person would do in the same situation. For instance, if you were asked to find out whether Lincoln ever visited Richmond, Virginia, and you did not know the answer, how would you proceed? First, it would seem reasonable to seek out biographies of Lincoln. Once you had one or more of them, it would be advantageous to search in the indexes under "Richmond", "Virginia", and the like. If there were no indexes, you might look through tables of content and section headings. Finally, you could just start reading the biographies. This suggests how the RS would proceed with text searches: examine RWIs to find pertinent documents; examine the indexes of relevant titles to narrow the inquiry; if indexes are lacking, examine the tables of contents; then, if there are any, examine individual section headings; finally, if still unsuccessful, read the books. Since an AIM system can read and understand English, we suppose that it has no trouble finding out if Lincoln "visited" Richmond, even if a biography states that he "went to" Richmond and never uses the word "visit".

To do an implied numeric calculation, such as the one implied by the sentence graphed in Figure 3, an AIM system will have to analyze the query and recast it as: "What was the difference in distance between the North American Plate and the Pacific Plate at the beginning of 1988 and the end of 1988". That is, "in 1988" can be interpreted in this context as "at the beginning of 1988 and the end of 1988"; and "How much did the North American Plate move with respect to the Pacific Plate" is interpreted as "What is the difference in distance between the North American Plate and the Pacific Plate". Recasting a query is effected by RSs that use natural language definitions and synonyms to go from the query to an interpretation form that can be used by the TDs. Now, before the distances can be calculated, the AIM system must, among other things, access the plate location data. In this instance, just to find out what the relevant site names on the two plates are, an RS will end up requesting a TD (in, of course, the relational logic natural language formalism) to "List all the western sites on the North American Plate" and "List all the eastern sites on the Pacific Plate". The TD, we suppose, will transform the requests into something usable by a particular database ED. Once the site names are known, an RS can ask a TD to fetch the data values for the appropriate time period; and so forth, until the user's implied calculation is satisfied.

In short, what we require of an AIM system is that it have sufficient RSs and TDs to make effective use of the documents in its docuverse. Clearly, this is not a trivial task and must be the subject of further research. We can see, however, what the high-level architecture of an AIM system must be like. It will have the basic architecture given in Figure 5.

5.0 Conclusion. Integrating diverse information sources and application software in a principled and general manner will require a very capable AIM system. In particular, such a system will need a comprehensive addressing scheme to locate the materials in its docuverse. It will also need an NLP system of great sophistication. It seems to me that the NLP system must serve three functions. First, it provides an NLI for the users. Second, it serves as the core component that understands and makes use of the RWIs contained in the docuverse. Third, it enables RSs to arrive at conclusions that can be transformed into procedures that will satisfy the users' requests. The best candidate for an "intelligent agent" that can satisfactorily make use of RSs and TDs appears to be an OODB. OODBs have, apparently, an inherent capacity to use the large numbers of RSs and TDs that will be required by an AIM system and an inherent capacity to use them in an effective way.

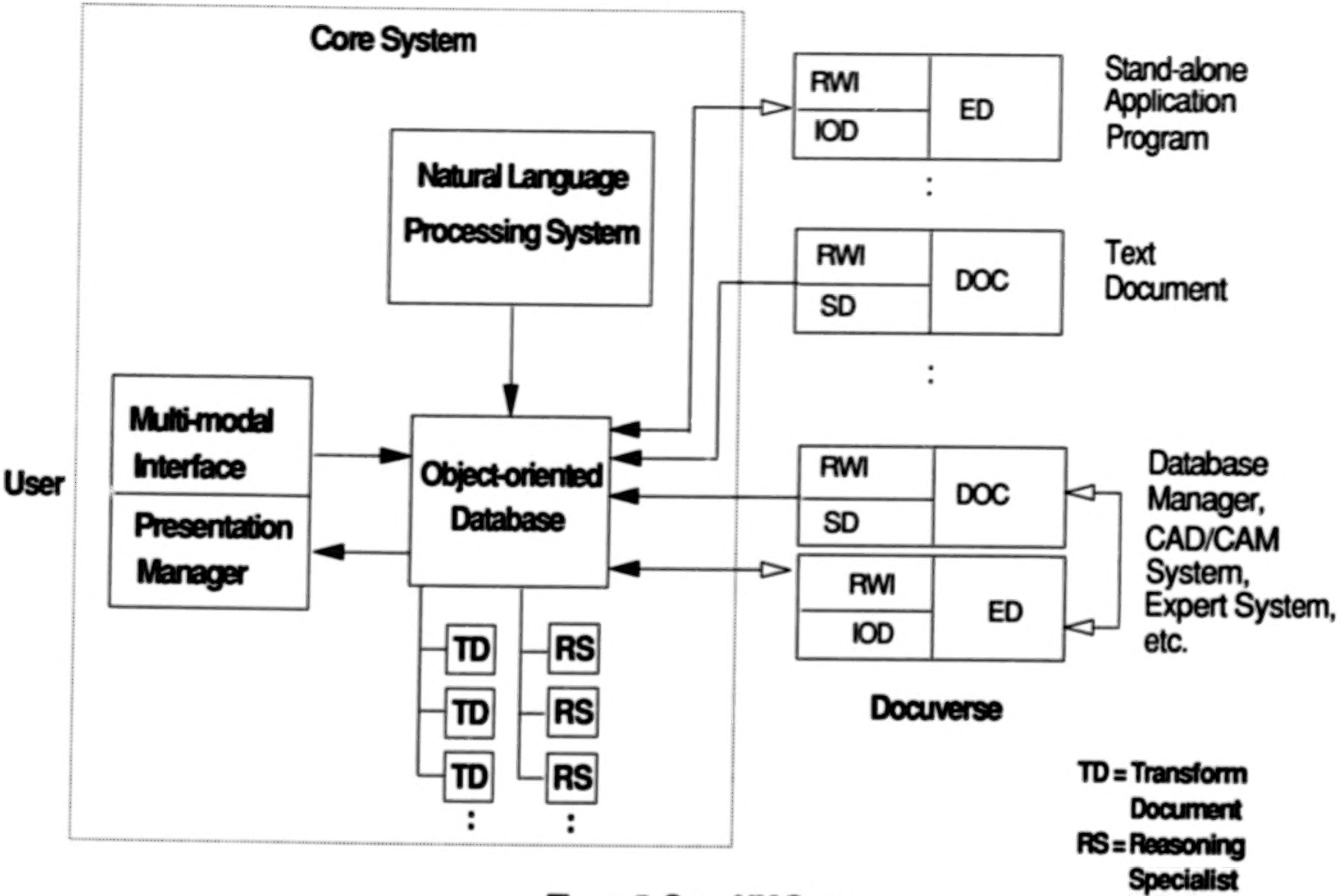

Core System
Natural Language Processing System
Multi-modal Interface
Presentation Manager
Object-oriented Database
User
TD
TD
TD
RS
RS
RS
RWI
IOD
ED
Stand-alone Application Program
RWI
SD
DOC
Text Document
RWI
SD
DOC
RWI
IOD
ED
Database Manager, CAD/CAM System, Expert System, etc.
Docuverse
TD = Transform Document
RS = Reasoning Specialist

Figure 5. Core AIM System

Bibliography

1. Nelson, T. 1988. Managing immense storage. BYTE, 13,no.1.225-238.
2. Short, N., W. Campbell, L. Roelofs, and S. Wattawa. 1987. The crustal dynamics intelligent user interface anthology. NASA technical memorandum 100693.
3. Baum, L. et al. 1987. The Erasmus system. Presented at the AAAI Blackboard Systems Workshop, Seattle, July, 1987.
4. Gazdar, G., E. Klein, G. Pullum, and I. Sag. 1985. Generalized phrase structure grammar. Oxford: Basil Blackwell and Cambridge, MA: Harvard U. Press.
5. Harrison, P. 1988. A New Algorithm for parsing generalized phrase structure grammars. Ph.D. dissertation, U. Washington, Seattle.
6. Fillmore, C. 1968. The case for case. In E. Bach and R. Harms, eds., Universals in linguistic theory. New York: Holt, Rinehart, and Winston.
7. Chafe, W. 1970. Meaning and the structure of language. Chicago: U. Chicago Press.
8. Quirk, R., S. Greenbaum, G. Leech, and J. Svartvik. 1985. A comprehensive grammar of the English language. New York: Longman.
9. Hudson, R. 1984. Word grammar. Oxford: Basil Blackwell.
10. Langacker, R. 1987. Foundations of cognitive grammar; volume 1, theoretical prerequisites. Stanford: Stanford U. Press.
11. Hoard, J. 1986. Drawing natural language inferences. Presented at the 61st annual meeting of the Linguistic Society of America, New York, December, 1986.
12. Barwise, J. and J. Perry. 1983. Situations and attitudes. Cambridge, MA: MIT Press.
13. Pollard, C. and I. Sag. 1987. An information-based syntax and semantics. Stanford: Center for the Study of Language and Information.
14. de Haan, J. and L. Schubert. 1986. Inference in a topically organized semantic net. Proceedings of the 5th National Conference on Artificial Intelligence (AAAI-86), 334-339.
15. Miller, S. and L. Schubert. 1988. Using specialists to accelerate general reasoning. Proceedings of the 7th National Conference on Artificial Intelligence (AAAI-88), 161-165.
16. Hurum, S. 1988. Handling scope ambiguities in English. Proceedings of the Second Conference on Applied Natural Language Processing, 58- 65. Austin, TX.
17. Miller, S. and L. Schubert. 1988. Time revisited. Proceedings of the Seventh Biennial Conference of the Canadian Society for Computational Studies of Intelligence, 39-45. Edmonton, Alberta, June, 1988.
18. Reinhart, T. 1983. Anaphora and semantic interpretation. Chicago: U. Chicago Press.
19. Bosch, P. 1983. Agreement and anaphora: A study of the roles of pronouns in syntax and discourse. New York: Academic Press.
20. Reichman, R. 1985. Getting computers to talk like you and me. Cambridge, MA: MIT Press.
21. Webber, B. 1988. Discourse deixis: Reference to discourse segments. Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics, 113-122.
22. Purdy, A., B. Schuchardt, and D. Maier. 1987. Integrating an object server with other worlds. ACM Transactions on Office Information Systems, 5:1.27-47.

BLANK PAGE

Modeling and Simulation

BLANK PAGE

A LOGICAL MODEL OF COOPERATING RULE-BASED SYSTEMS

Sidney C. Bailin
John M. Moore
Robert H. Hilberg
Elizabeth D. Murphy
Shari-Ann Bahder

CTA INCORPORATED
14900 Sweitzer Lane
Laurel, MD 20707

1 Introduction

This paper describes an abstract model of cooperating rule-based systems (CRBSs). Rule-based systems are expected to play an increasingly important role in space missions of the 1990s and beyond. As the trend towards distributed operations continues, specialized RBSs will have to share information and perform in a coordinated manner to ensure safe and economical overall mission operation.

The ultimate goal of this research is to define a *formal* model. Such a model will characterize the elements and operations of cooperating rule-based systems in the form of abstract data types. This would provide system planners with a precise language for specifying requirements. It would provide developers with a reusable set of package specifications for CRBSs.

The model presented in this paper is a step in the direction of a formal model. The current model may be regarded as *semi-formal*. It describes a layering of CRBSs elements and operations, and is based on the idea of peer-to-peer communications adopted from the Open System Interconnection model [1].

To arrive at an understanding of the problems of cooperating rule-based systems, we selected a specific application for analysis, and then tried to extrapolate to the more general case. Section Two contains our analysis of CRBSs in Payload Operations and Network Control Centers.

Section Three abstracts from the analysis of Control Centers in order to identify generic requirements of cooperating rule-based systems. We identify six high-level requirements, which together characterize the problems inherent in cooperation among intelligent, autonomous agents.

Section Four presents the logical model. The model provides an organized way to view both the requirements and potential technical solutions.

Sections Five and Six summarize our most recent work, which has focused on identifying the reasoning capabilities required of cooperating rule-based systems. While the material of the first four sections appeared in an earlier version of this paper [2], the material of Sections Five and Six is new. We have removed details included in the earlier version in order to highlight the essential points of our analysis. We take advantage of the resulting space saving by including our latest results.

2 Cooperating Rule-Based Systems in Control Centers

2.1 Functions Performed in Control Centers

This section describes the functions of a Payload Operations Control Center (POCC) and of a Network Communications Center (NCC), as a basis for identifying potential roles of rule-based systems.

Payload Operations Control Center. The primary responsibility of a POCC is to prepare automated procedures for controlling spacecraft and instrumentation during flight, and to monitor the health and safety of the spacecraft and its components. A list of functions performed by a

This work was supported by NASA's System Autonomy Program through contracts NAS5-27684 and NAS5-31500.

POCC is contained in Table 1. The functions performed by a POCC can be divided into three major areas: command management, telemetry processing, and interfacing with other facilities and users. In addition to these major categories of functions, a POCC must also provide for spacecraft and operations simulation and for allocating resources for data operations control of on-line resources.

The command management function consists of planning the payload and sensor activities that will be performed during the mission, developing the command sequences that will effect these activities, verifying that the commands are correct and will not violate operating constraints, developing mission schedules, and keeping a history of the commands that have been carried out by the payload.

The telemetry processing function consists of receiving and recording telemetry data and then performing a variety of tasks using these data. Health and safety data are monitored and data quality checks are performed. Experiment data are reviewed to ensure the validity of the data being received. Attitude data are stripped from the data stream and formatted for transmission to Flight Dynamics.

The POCC must interface with other facilities and users on a continuing basis. Commands are received from the Command Management System, and attitude data are sent to Flight Dynamics. Depending on the mission, frequent communications with experimenters and users may be needed. The interface with the NCC must be maintained so that the spacecraft can be controlled during data communications contacts.

The above functions are normally performed by four controllers. The Payload Operations Controller (POC) conducts all spacecraft contacts and makes decisions on the control of the observatory. He also coordinates with the experimenters through the Experiment Operations Facility (EOF). The Observatory Engineer (OE) is responsible for monitoring, maintaining and approving the performance of the observatory. The Instrument Engineer is responsible for monitoring, maintaining and approving the performance of the instruments.

Table 1: Payload Operations Control Center Functions

- Command Management
    - Command Processing
    - Command Verification
    - Payload Operations and Control
    - Instrument and Sensor Operations and Control
    - Mission Planning and Scheduling
    - Command History Generation
- Resource Allocation
- Spacecraft/Operations Simulation
- Telemetry Processing
    - Experiment Quick Look Processing
    - Monitoring Spacecraft and Instrument Health and Safety
    - Strip and Format Attitude Data
    - Receive and Record Telemetry
    - Generate Data Presentations
    - Check Data Quality
    - Data Management
- Interfacing with Other Facilities and Users
    - Receive Commands from Command Management System
    - Send Attitude Data to Flight Dynamics
    - Interface with Experimenters
    - Interface with NCC
    - Control Spacecraft Contacts

The Ground Controller (GC) is responsible for configuring, coordinating, and operating the ground system. The GC coordinates the support required from all ground facilities in planning operations. In addition to these control positions in the Mission Operations Room, there is a Command Management Engineer (CME) in the POCC who works with experimenters and operations personnel in the orbit-by-orbit mission planning and contact message generation.

While almost every function itemized above is supported by a variety of computers, there is heavy dependence on the manual control by human operators in carrying out these activities. Even where automated sequences are stored to perform a function, the sequence is usually operator initiated. These functions depending on manual control are the areas for which rule-based systems have the potential for enhancing system performance and efficiency.

Network Communications Center (NCC). The primary function of a network communications center is to act as an interface between network resources and the users of those resources. In a satellite tracking network, the network communications center ensures that the scheduling of network elements comes as close as possible to satisfying user event requirements while remaining within system integrity constraints. To meet these conditions, the network communications center must perform various lower-level functions, as indicated in Table 2. In performing these functions, the network communications center seeks to maximize utilization of network resources and services.

At current levels of automation, the NCC's scheduling functions are largely performed by scheduling algorithms. Schedule optimization and conflict resolution require rule-based reasoning, which is performed by operators at the Schedule Generation (SG) console position. SG operators also monitor and control the activation and transmission of the final schedule to users and network elements. The Active Schedule (AS) operator monitors and controls the active schedule process, handling requests for real-time changes to the schedule due to spacecraft emergencies or unexpected opportunities for scientific observation.

Table 2: Functions of a Network Communications Center

- Provide pre-launch and operational support to network users
    - Perform periodic system testing
    - Perform spacecraft simulation
    - Process requests for performance data
    - Monitor real-time events
    - Provide post-event reports and problem reports to users
- Allocate network resources according to established user priorities
    - Perform routine network scheduling
    - Perform emergency scheduling
    - Process near real-time and real-time requests for service reconfigurations
- Coordinate and control network activities
    - Monitor network site status
    - Analyze network performance
    - Perform network fault isolation
    - Direct corrective actions
    - Report to network elements on network performance

During the real-time support period, the NCC monitors network performance data and controls the network by issuing real-time control messages. Operators at two console positions continuously monitor the network's performance. Actual performance data are compared with predicted performance data to identify and locate system failures and degradations. When an anomaly is detected, these operators perform rule-based fault isolation and, with their supervisors, determine the NCC's response. The NCC's responsibility is to identify the cause of the anomaly and to direct corrective action.

Real-time control takes the form of service reconfiguration messages that are issued to ensure uninterrupted receipt of user telemetry. Service reconfigurations may be initiated by the user or by the Operations Controller, who monitors and controls real-time support.

The NCC provides pre-launch support to users in the form of spacecraft simulation. Data about each spacecraft, including event histories, are stored in the NCC databases. At predefined intervals, the NCC performs routine verification tests to ensure network readiness. The NCC also maintains files on network site status in order to support timely service reconfigurations in the event of equipment failures. The Performance Analyst (PA) is responsible for maintaining an updated site-status display and an understanding of network status, as a basis for performing fault isolation. Both the OC and the PA coordinate frequently with the scheduling operators. Network performance is summarized in periodic reports issued by NCC management.

2.2 Rule-Based Systems in Control Centers

We have identified five types of rule-based systems that could be used in control centers: data analysis, scheduling, information retrieval, diagnostics, and simulation. In interfacing with the user, rule-based data analysis and scheduling systems could assist in checking parameter compatibility, in advising adjustments where possible, and in looking for resource allocation conflicts between the newly-defined task and previously-planned tasks. Rule-based information retrieval systems could be useful in looking up previous results from similar events, especially if the data are stored in multiple, dissimilar databases.

While looking up task specifications may be straightforward, locating other applicable data requires an understanding of the information in the available databases and the correct means of accessing the information. This level of understanding suggests a requirement for rule-based reasoning. Task scheduling/allocation and system monitoring/adjustment require the ability to reason about the task requirements and about the current system status. Rule-based schedulers and diagnostic systems have been used for such purposes.

Potential applications of rule-based systems in the mission planning and scheduling area include task scheduling, simulation, and information retrieval. The scheduler must work within the limits set by mission constraints. Its primary goals (in order of importance) would be to guarantee the safety of systems under its control, to find work-arounds to constraint violations, and to generate a schedule that is nearly optimal. Both the scheduler and the diagnostic system must be able to respond to the rapidly changing states of the systems under their control and still provide near-optimal performance. To accomplish this, both systems will have to work in near real time.

Using these types of rule-based systems, we have derived a high level partition of the functions performed in a control center. This is shown in Figure 3. (Reference [3] describes a different approach.) Requirements in terms of a scientific program, a requested event, or a query enter the system from a customer. The controller passes information concerning requested events to the **scheduling system**, and information concerning queries to the **diagnostic system**. If additional information is required to process the request, it is found and presented by the **information storage and retrieval system**. The **information storage and retrieval system** is one of the prime agents for effecting cooperation among systems. Data regarding environmental status, subsystem status, and schedule are essential for any of the RBS components to perform properly. If simulations are required to process the request, then these are performed by the **simulation system**. The results of any simulations must be fed back both to controllers and system elements such as the **diagnostic system** and **scheduling system**. It may also be necessary for the **scheduling system** and the **diagnostic system** to communicate to obtain further information. In this case, information from the **information storage and retrieval system** is also likely to be needed. It may be necessary for the controller to perform **data analysis** to relate the request to the capabilities of the control center. Similarly, it may be necessary for the customer to perform **data analysis** to derive requirements or relate responses to a particular problem being addressed.

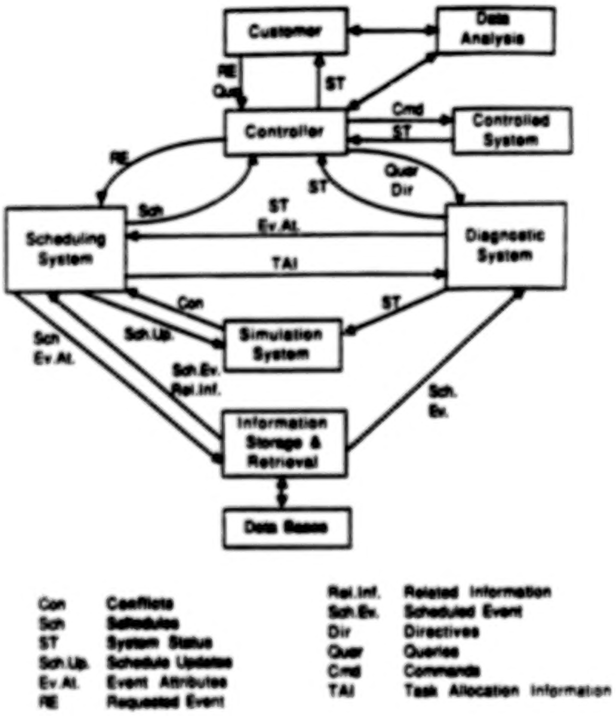

Customer
Data Analysis
RE
Quer
ST
Controller
Cmd
ST
Controlled System
RE
Sch
ST
Quer
Dir
ST
Ev.At.
Scheduling System
TAI
Diagnostic System
Con
ST
Sch.Up.
Sch
Ev.At.
Simulation System
Sch.Ev.
Rel.Inf.
Sch.
Ev.
Information Storage & Retrieval
Data Bases
Con Conflicts
Sch Schedules
ST System Status
Sch.Up. Schedule Updates
Ev.At. Event Attributes
RE Requested Event
Rel.Inf. Related Information
Sch.Ev. Scheduled Event
Dir Directives
Quer Queries
Cmd Commands
TAI Task Allocation Information

Figure 3: Flow of Information in a Control Center

BEST COPY AVAILABLE

Two conclusions emerge from this discussion of the interactions between RBSs: 1) the information passed between elements must be current, and this information may be frequently updated by yet a third system element; 2) there is a need for the system to learn, since it is impossible to foresee every situation that may occur, and also because commands that had a particular meaning at the beginning of a mission may produce different results after a component, such as the spacecraft solar array, has degraded over a number of years.

3 Generic Requirements of Cooperating Rule-Based Systems

From the description of RBS interaction in Control Centers, we identify the following high-level requirements which are intrinsic to the problem of cooperation:

- Partition knowledge
- Model other agents
- Communicate with other nodes
- Handle differences in concept repertory
- Define and allocate work
- Coordinate activities

3.1 Partition Knowledge

We have seen that a logical partitioning of knowledge in a ground control environment produces five kinds of rule-bases. Partitioning a knowledge base can yield both increased performance through parallelism, and enhanced extensibility and/or maintainability of the knowledge base (references [4] and [5] provide useful background on this topic).

Each partition requires a knowledge structure that makes explicit the most important aspects of information in the partition. For example, knowledge allocated to a rule-based diagnostic system may be best represented as a model of the system being controlled. Knowledge allocated to a rule-based scheduling system might best be structured in terms of event models.

The knowledge in several partitions may overlap. Some overlap is necessary for the nodes to communicate, and it may be necessary for nodes to use the same representations for overlapping knowledge areas. A cooperating rule-based scheduler and rule-based diagnostic system might, for example, use the same schedule and system status knowledge. Each system would specialize in just one of these two categories, but they would share both types of knowledge between themselves.

An alternative to providing a common representation for overlapping knowledge is to map information between representations and partitions. Because the effectiveness of each node depends on its knowledge and knowledge structures, use of a common representation may not be appropriate. For example, the different knowledge structures employed by the diagnostic system and scheduler (i.e., system model *vs.* event model) might necessitate mapping shared status and schedule information between these representations.

It may be useful to provide knowledge partitions with a means of learning new rules. This may be necessary in domains where problems gradually increase in difficulty. Rule-based diagnostic systems, for example, could mimic the development of human expertise over time if they could learn new mappings between symptoms and causes of system anomalies.

The ability to learn implies a requirement to verify and incorporate new knowledge. A new rule may not be consistent with existing rules, or it may only reflect the generating node's own interests and ignore global CRBS problem-solving constraints. Thus, any new rules must be verified before they are added to a node's rule base. New rules may change the relationships between existing rules. For instance, a new rule may alter the order in which the node's existing rules are applied to a problem. Such changes must be accounted for when the new rule is incorporated into the node's rule base. This may involve modifying existing rules.

Another potential requirement is the ability to abstract over a knowledge structure and the information in that structure. This requirement may be particularly relevant in hierarchical control, when the controlling node needs an abstraction of its subordinate nodes' status and understanding. For example, if a rule-based diagnostic system uses a hierarchical model of the system being controlled, there will be a requirement for abstraction of symptoms between levels of the hierarchy.

3.2 Model Other Agents

The requirement to model other agents is derived from the need to partition knowledge. In order to work together, component RBSs must be aware of each others' existence and attributes. The models must represent the partition of knowledge and the inter-partition mappings in a way that is accessible to the RBS reasoning facilities. It is especially important to model the skills and goals of other RBSs. Skills describe the capabilities of a node's rule-base. Goals describe in a general way how the node will use its skills.

3.3 Communicate With Other Nodes

Cooperation between RBS presumes that the systems communicate. The types of information transferred between nodes may vary from simple status values to entire chains of reasoning, including hypotheses and associated assumptions, and even rules. Cooperation between the five types of rule-bases we have described would require at least the transfer of status information, event schedules, diagnostic results, data-analytic results, and simulation parameters/results.

3.4 Handle Differences in Concept Repertory

Each cooperating RBS needs criteria for evaluating information it receives from other RBSs in terms of importance, certainty, and timeliness. Irrelevant, uncertain, or out-of-date information may distract an RBS from promising lines of reasoning. Failure to send/accept information that is relevant, reasonably certain, and up-to-date may cause an RBS to overlook potentially important lines of reasoning.

Received information may be inconsistent with prior knowledge. For example, the scheduling information about a spacecraft event that has been reconfigured in real time may appear to be inconsistent with the diagnostic system's model of the spacecraft. Such conflicts must be identified and resolved, perhaps jointly by the sender and receiver.

Depending on the knowledge partition, two communicating RBSs may work on different levels of abstraction or on different projections of the information passing between them. Abstracted information is useful in hierarchical control schemes, i.e., when a higher-level controller is conerned with general trends but not with all details about the controlled system. Different projections are required when two nodes work with different structures obtained from the same data. The data analysis and information retrieval RBSs, for example, would project the same data onto different structures. The information retrieval system is concerned with control and classification structures by which the data are stored and accessed. The data analysis system is concerned with values *within* the data.

3.5 Define and Allocate Work

Partitioning knowledge does not in itself accomplish anything for an application. If RBSs are to work productively together, there is a requirement to partition not just knowledge but also the work to be done. The first step is to decompose the overall system task into subproblems. The decomposition must not impose unhealthy interactions on the nodes (e.g., the resource usage implied by the decomposition should not endanger the system), and it should facilitate the integration of results into an overall solution. In the Control Centers, problem decomposition should be straightforward since the roles of each RBS are well defined and distinct. For example, if the scheduler discovers that it does not have all the event parameters that it needs to create a schedule, it should know to ask the information storage and retrieval system to retrieve them.

If the cooperating RBSs have different capabilities, it may be necessary to determine which RBS is best suited to solving each task or subproblem. Skill information stored in the node models can be used in these decisions. For instance, the Control Center scheduler might use its understanding of the simulator and diagnostic system in determining which of these is best qualified to evaluate a proposed schedule (or partial schedule).

There must be a way of describing subproblems to other nodes. In the Control Center RBSs, the data analysis system must be capable of specifying scheduling and retrieval tasks to the scheduling and information retrieval systems. The capability to specify work is complicated by the fact that nodes may use different knowledge to interpret a work specification, and they may have conflicting subgoals. Node models may help in determining how a specification should be presented. Scheduling and control mechanisms are required to ensure that nodes process their assigned subproblems correctly.

Nodes must have a way of identifying unsolvable subproblems and a way of negotiating subproblem details. A node may determine that an assigned subproblem is unsolvable, or it may require more details before accepting the work. For example, the control center RBSs may easily specify an erroneous query to the information retrieval RBS. It is the retrieval system's responsibility to recognize such errors and determine in conjunction with the requestors what information they really need.

More than one node may appear qualified to solve a subproblem. Bids may be accepted to resolve such conflicts. This type of behavior is not currently exhibited in the automated parts of control centers (it is performed by human supervisers), but it could prove useful in implementing a hierachical scheduling system, for example. Individual nodes could be designed as experts at scheduling particular types of events, and they could bid for work when such events become available.

The cooperating RBSs may experience changes in load as a result of processor/node failures or changes in the stages of a mission. The system must be able to respond to such changes effectively. In cases where failed nodes were performing critical work, the system must (if possible) reallocate the work to other nodes.

3.6 Coordinate Activities

Partitioning work ensures that each RBS knows what it is supposed to do within the overall system. Models of other nodes and communications between them ensure that the RBSs can interact in performing their tasks. Synthesis of the subtasks into a global objective requires more than this, however. The RBSs must not only communicate, they must coordinate their activities. Scheduling and control mechanisms are required to ensure that the necessary degree of coordination is maintained.

A basic requirement is to ensure that the individual RBSs do not work at cross purposes. RBSs may cooperate by informing each other of their beliefs and intentions. Nodes must have some basis for deciding when to send such information, such as their goals, to other nodes. The timing or scheduling of communications could, for example, be linked to the development of system products or results. Other criteria could be formulated as problem-specific heuristics.

Ensuring coherent progress towards system objectives may not be sufficient, as the solutions found may have to be optimal. For example, the scheduling of network resources should be optimized. The control and coordination strategy must be capable of matching each subproblem to the most appropriate RBS.

While the coordination mechanisms have to ensure that convergence is reached, they must also guarantee that the overall system is not jeopardized by the node's interactions. Ground control centers have strict constraints on resource use: control strategies must avoid jeopardizing resources or RBSs, deadlocks over coordination and use of resources must be avoided, and resource allocations must be determined. These requirements imply a more general requirement: there must be a means of synchronizing the RBSs' use of resources.

System objectives may suggest either distributed or centralized control, or both. Centralized control is desirable when some node interactions are dangerous. A centralized controller can guarantee that such interactions do not occur. Distributed control may be useful when there are stringent time constraints because it frees the RBSs from having to wait for a centralized controller to authorize or schedule actions. It does, however, complicate the coordination process. In control centers, distributed control would be appropriate for non-emergency interaction between the scheduling, diagnostic, and data analysis RBSs. During an emergency some form of centralized control is required in order to prevent damage to the system. Such control is currently performed by human supervisers. The change in control strategies as a response to system emergencies suggests a more general requirement: cooperating rule-based systems must be capable of adjusting their control strategy in response to changes in their environment.

Both peer-to-peer and hierarchical control relationships may exist between RBSs. The control center scheduling and diagnostic systems work in parallel, for example. Interaction between the scheduler and information storage/retrieval system is hierarchical: the scheduler issues commands to the retrieval system. Peer RBSs may seek to maintain mutual consistency of their actions, or they may simply share information. If control is hierarchical, the controller may constantly monitor and send directives to a controlled node, or it may send periodic directives. All of these control strategies may be required in the same system.

In some cases, it may be desirable to consolidate computational resources to work on subproblems. For example, sub-RBSs in a diagnostic system might work together on part of the diagnosis. While members of the group work on their common subproblem, other RBSs should be able to address the group as a single unit. Grouping RBSs can greatly reduce the number of interactions that must be modeled and managed.

4 The Logical Model

The logical model provides a framework for discussing solutions to unique CRBS requirements. As illustrated in Figure 4, the basic model consists of four layers, each of which enriches the knowledge available at the lower layers. The layering of capabilities offers the following potential benefits:

- A basis for planning CRBS evolution
- A precise means of specifying or assessing CRBS capabilities
- Modularity in the design of CRBSs

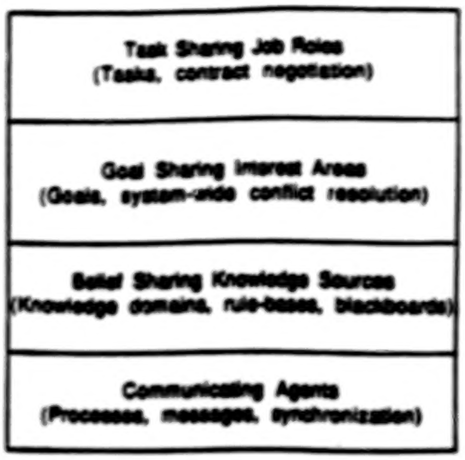

Task Sharing Job Roles
(Tasks, contract negotiation)
Goal Sharing Interest Areas
(Goals, system-wide conflict resolution)
Belief Sharing Knowledge Sources
(Knowledge domains, rule-bases, blackboards)
Communicating Agents
(Processes, messages, synchronization)

Figure 4: The Basic Model Consists of Four Layers

CRBSs may be designed to incorporate some of the layers or all of them, and may be classified on the basis of the layers provided. Since the higher layers provide a greater degree of cooperation between nodes, the layering can be used to plan the evolution of CRBSs from a basic capability to a more powerful one. The layers may also be used to specify the degree of cooperation (and consequent intelligence) required of a new system of CRBSs, and to assess the degree of cooperation and intelligence in existing systems.

The lowest layer refers to a network of communicating processes, which we call *agents*. At this level we make no assumptions of intelligence or rule-based capabilities, nor of the purposes of communication between agents. The agents may communicate data of any type. No distinctions are made between transient data, domain knowledge, and meta-knowledge.

The second layer introduces the idea of rule-bases. A *knowledge source*, according to our definition, is a rule-based communicating agent specializing in some knowledge domain. Knowledge sources communicate by sharing *beliefs*. Beliefs are any data that are yielded by the inference process, e.g., hypotheses being

BEST COPY AVAILABLE

pursued or facts that have been established. At this layer, we assume that knowledge sources communicate in order to enrich their respective knowledge, but we make no other assumptions concerning the content or means of communication.

The third layer introduces the idea of *interest areas*, which are the first indication of goal-directed reasoning in the model. An interest area is a set of goals that motivate and guide the reasoning process. An interest area determines the way knowledge is applied. Interest areas interact by exerting influence over each other. The principal means of exerting such influence is through the communication of meta-level knowledge such as goals, priorities, and plans.

The fourth layer introduces the concepts of tasks and job roles. This is the layer at which cooperation is most fully achieved. Overall system goals are decomposed into tasks, which are allocated among various job roles. The definition and allocation of tasks may be static or dynamic, or some combination of the two. Static tasking is accomplished either through system design decisions (i.e., component RBSs are assigned specific parts of the system task) or through tables that may be updated periodically. Dynamic tasking is performed by the run-time system, and involves identifying, decomposing, and assigning tasks based on the current system state.

4.1 An Evolutionary View of the Layers

We can view the development of CRBSs as a staged process in which each stage provides a new layer of the model. In the first stage, effort would be focused on developing a network of communicating nodes. Issues such as the communications medium, connectivity, protocols, and operating systems must be addressed. The constituent nodes might be rule-based systems whose designs indirectly impose constraints on communications. Such influence would be exerted through the external and program interfaces of each RBS. The communication and compatibility issues addressed would not, however, be formulated in terms of rule-based processing, but in terms of conventional inter-process communications.

In the second stage of CRBS development, attention would be focused on providing the constituent RBSs with a means of sharing knowledge. Blackboard technology might be introduced at this level. Issues of knowledge partitioning, overlap, representation, and translation have to be addressed. Belief sharing protocols have to be established and implemented in the constituent RBSs. Such protocols would define the conditions under which beliefs are to be shared with other nodes, and the criteria for evaluating and incorporating received knowledge.

The third stage of development would introduce system-wide conflict resolution mechanisms, i.e., mechanisms for choosing among alternative lines of reasoning. Each RBS may already incorporate some form of conflict resolution as a standalone system. The mechanisms introduced at this stage, however, would enable the constituent RBSs to influence *each other's* choices of a reasoning path. Knowledge sharing issues analogous to those considered in the second stage would have to be addressed again, but with a different focus. While in the second stage the emphasis was on sharing beliefs that refer directly to the application domain, in this third stage the emphasis is on meta-level knowledge that can be used to direct the reasoning path.

During the first three stages of development, there is an implicit decomposition of work to be performed by the different RBSs. The decomposition and allocation of work is implicit in the content of the rule-bases, and in whatever procedural logic is built into the consitutent RBSs. In the final stage of CRBS development we would systematize the partition of work, either by making explicit the existing static partition, or by introducing dynamic mechanisms for defining and allocating tasks. The benefit of making a static partition explicit is that the partition becomes easier to modify. For example, if in stage three an RBS's assignment was implied by that system's rules and procedural logic, the fourth stage might involve developing truth maintenance procedures that facilitate the addition of new rules or other modifications of the rule-base. Capability tables could provide a basis for implementing dynamic contract negotiation and task assignment protocols.

5 Required Reasoning Capabilities

The purpose of this section is to characterize the required reasoning capabilities of cooperating rule-based systems. Such a characterization refines our understanding of the functions of CRBSs, and thus contributes to our long-term goal of a formal model. From the analysis of the preceding sections, we derive the following high-level logical requirements:

- Handle uncertainty
- Reason temporally
- Abstract features of a problem or situation
- Plan
- Learn
- Model other agents
- Explain beliefs, decisions, etc.

5.1 Handle Uncertainty

Cooperating rule-based systems must be capable of expressing a belief as a belief rather than a fact. Uncertainty is a potentially a factor in all of the Control Center rule-based systems described in Section Two. In the generic scenario of cooperating rule-based systems, a key source of uncertainty is the distribution of work among nodes with different functions, viewpoints, etc.

The following specific capabilities are needed to handle uncertainty:

- Express/recognize uncertainty
- Reason tentatively
- Change beliefs

Handling uncertainty is principally a question of managing beliefs. The requirement to handle uncertainty applies, therefore, to the level of belief-sharing knowledge sources (layer 2). A knowledge source communicating beliefs to its peers may want to qualify the beliefs with an expression of uncertainty. The degree of certainty may reflect the plausibility of the belief, the amount of evidence that exists for the belief, the assumptions that underly the belief, or other measures. The receiving knowledge source must be able to recognize and understand this expression of uncertainty.

Communicating degrees of certainty serves no purpose, however, if the relative certainty of a belief has no impact on the inferences drawn from the belief. Belief sharing knowledge sources must be able to incorporate uncertainty into the reasoning process, i.e., to reason tentatively. They must apply some method of compounding or combining uncertainty when inferences are drawn from more than one uncertain belief. For example, Bayesian reasoning assigns real-valued probabilities between 0 and 1 to assertions. The logical conjunction of two assertions is assigned the product of the probabilities of the component assertions.

The capability to reason tentatively implies that beliefs will sometimes have to be retracted or altered. Retraction of a single belief may invalidate many (but possibly not all) consequences of the belief. The knowledge source must be capable of determining which consequences need to be retracted. Besides retracting beliefs, the degree of certainty assigned to a belief may increase or decrease (or change in some other way, if a non-numerical measure of certainty is being used). The knowledge source must have a means of determining the impact of such changes on related beliefs, e.g., on consequences of the given belief, and on alternatives to it.

5.2 Reason Temporally

Time is an important factor in virtually all space information system environments. Systems must be able to respond to changes of state, to operate within time constraints, and to manage interdependencies between events. Rule-based systems must therefore be capable of reasoning about dynamic states, time constraints, and interdependencies. In Control Centers, the requirements for temporal reasoning are found in the rule-based systems with the most dynamic environments, i.e., the scheduler and diagnostic systems. In the generic model, requirements for temporal reasoning arise as a result of dynamic interaction between agents.

Our analysis of the need for temporal reasoning leads to the following detailed requirements:

- Apply temporal qualifiers to beliefs
- Draw conclusions from temporally qualified beliefs
- Update beliefs based on time changes
- Reason about time constraints

The first three detailed requirements concern the qualification and manipulation of beliefs, and therefore apply to the level of belief sharing knowledge sources. Temporal qualifiers are conceptually similar to the certainty qualifiers mentioned in Section 5.1. Temporal qualifiers may describe the time period(s) over which an assertion is believed to be true, e.g., always, periodically, in the past, for a specified interval, etc. Temporal qualifiers may also describe relationships between events, e.g., event **B** follows event **A**.

Reasoning about time constraints is potentially a requirement at the three upper layers of the logical model: belief sharing knoweldge sources, goal sharing interest areas, and task sharing job roles. Knowledge sources must be aware of time constraints in order to recognize state changes that occur when various intervals elapse, and also to be able to recognize temporally contradictory assertions. Interest areas must be able to reason about time constraints in assessing the feasibility of goals and/or the effectiveness of strategies. Task sharing job roles must be able to reason about time constraints in decomposing, assigning, and monitoring the progress of work.

5.3 Abstract Features of a Problem or Situation

The capability to abstract is required whenever a hierarchical control scheme is applied to solving a problem. Higher layers in the hierarchy must ignore details that are the strict concern of lower levels. Abstraction is also necessary when different agents take different views of a problem. It is necessary to understand the common features of alternative views in order to assess their consistency and/or their completeness. Abstraction is a required reasoning capability throughout the five types of Control Center rule-based systems, and in all six functional areas of the generic description. We conclude that abstraction, far from being a desirable but obscure capability, is a crucial logical requirement of cooperating rule-based systems.

The following specific capabilities will provide the needed abstractions:

- Classify objects, events, beliefs
- Identify important features
- Project onto selected attributes

A framework for abstraction is obtained by classifying objects, events, and beliefs into one or more inheritance hierarchies. Such hierarchies identify those attributes that are common among classes and those that may vary. Abstraction is the process of viewing an object, event, or belief as an element of an appropriate class. In order to select an appropriate class, an agent must determine the important features of the element, and those that can be ignored for the current purposes. Once the important features have been selected, the agent must be able to represent the element solely in terms of the important features, i.e., to project onto those features.

Belief sharing knowledge sources must be capable of classifying problem-domain objects, events, and beliefs. Interest areas must be capable of classifying higher-order objects as goals and plans, and events such as changes in goals and plans. Interest areas may need to classify beliefs in order to determine appropriate problem-solving strategies. Task sharing job roles must classify tasks, resources, problem-solving methods, potential task-execution problems, and status data. At each of these layers the active program must model, and possibly classify, its peers.

5.4 Plan

Planning is performed at the top two levels of the model, i.e., by goal sharing interest areas and task sharing job roles. Not surprisingly, the Control Center rule-based activity in which planning is most needed is scheduling. Among the high-level generic functions of cooperating rule-based systems, planning plays an essential role in defining and allocating work, and in coordinating activities.

The following steps describe a generic, cyclical planning function:

- Identify goals
- Develop plans
- Assess impact of other agents' goals and plans
- Measure progress
- Reassess and possibly change plans

Described in this manner, planning is a *function* rather than a mode of *reasoning* (such as, for example, temporal reasoning). Each step in planning process, however, is potentially defined largely in terms of rules. When we refer to planning as a required reasoning capability, we are referring to the logic of such rules.

5.5 Learn

The capability to learn is a key requirement of adaptable autonomous systems. It is the principal means of overcoming the limitations of knowledge engineering, i.e., the attempt to codify in a set of rules all possible situations with which the system will have to cope. The types of learning required range from the introduction of entirely new concepts into a system's knowledge base, to the addition of new beliefs or rules into the knowledge base, to the refinement of existing beliefs or rules.

New knowledge may be either received from outside the learning system, or generated from within the learning system. The term "machine learning" usually refers to the latter case, which involves an ongoing process of creating and refining hypotheses to account for known data. The case in which knowledge is received from the outside in known as "knowledge maintenance." In the case of human learning, the incorporation of new knowledge received from the outside is as important as the internal generation of hypotheses. In the human case, the "outside" may consist of books, other humans, etc. In the case of cooperating rule-based systems, the outside consists of a component system's users and neighboring systems.

Knowledge maintenance involves assessing the impact of new knowledge, e.g., determining whether it is consistent with previous knowledge, determining whether any rules are now redundant or unreachable, etc. There is a need for automated knowledge maintenance functions even when the source of the new knowledge is human. Experience has shown that humans are not effective in assessing the impact of new rules on a rule-base of, say, more than a hundred rules. When a rule-based system is sharing knowledge with a neighbor, the need for automated knowledge maintenance is even greater. We suggest that this is an important aspect of learning in cooperating rule-based systems. We therefore identify the following specific requirements for learning:

- Create hypotheses to account for available evidence
- Refine hypotheses
- Incorporate new concepts, facts, or rules
- Assess impact of revised beliefs/rules, e.g., on their consistency with other beliefs/rules

The distinction between learning beliefs and learning rules may be understood in terms of the layers of our model. Adding or refining beliefs will occur predominantly at the knowledge source level. Adding or refining rules for obtaining new knowlegge should be performed at the interest area level. This is because the application of such rules is subject to the conflict resolution and goal-directed strategies of the interest area level.

Both types of knowledge, however, may be acquired at any of the three upper layers of the model. For example, knowledge sources may develop meta-rules for knowledge maintenance, i.e., assessing the consistency of new beliefs with previous beliefs. An interest area might learn that a certain peer cannot be trusted to fulfill its declared goals: this is a case of acquiring a new belief, rather than a rule. At the job role level, a system may learn the new fact that a certain problem is unsolvable. It may also, however, generalize this fact to create a rule that rejects as unsolvable all problems in some class.

5.6 Model other Agents

Modeling the goals, plans, and capabilities of other agents is a knowledge representation problem whose solutions should draw heavily on structural approaches such as frames and semantic networks. There are, however, purely logical issues that must be addressed, such as the following:

- Identifying concepts used in common with other agents, or related to those of other agents
- Identifying relationships between beliefs of other agents and those of self
- Identify potential interaction between the goals and plans of other agents and those of self

Semantic networks provide a means of addressing the first of these requirements, but the latter two suggest a rule-based approach for maximum flexibility. Cooperating rule-based systems require logical constructs that facilitate the specification, evolution, and application of rules defining agent interrelationships at all layers of the logical model.

5.7 Provide Explanations

The crudest form of explanation is simply a dump of the inferences through which a conclusion was reached. Such explanations are probably not useful for online operations in a control center or similar environment. They may be interpretable by a batch processing program, or by a human working "offline." In such cases, the significant work lies in the interpretation—not in the explanation itself, which is simply a matter of bookkeeping.

A non-trival explanation capability must provide a *meaningful* explanation of a belief or decision. This might involve classifying a chain of inferences as an instance of a certain class of arguments. It might involve identifying key rules or items of information that were used to derive the belief or arrive at the decision. In addition to determining the content of the explanation, a useful facility must determine how to present the explanation in terms understood by the recipient. Thus, we have identified the following specific requirements:

- Present train of reasoning
- Present principle(s) justifying belief, decision, etc.
- Tailor explanation to the understanding of the recipient

For a rule-based system to provide explanations, it must be capable of reasoning to some extent about its own rules. This does not necessarily imply a need for higher-order logic—for example, any formal system containing elementary arithmetic can reason about its own rules by encoding them arithmetically. The interest area layer in our logical model appears to be the most appropriate location for an explanation facility, since this layer is responsible for guiding the application of rules by a knowledge source. An interest area could naturally provide explanations about its own behavior because the elements of discourse at that level are such things as goals, plans, strategies, etc. An interest area could not, however, provide explanations of decisions made at the job role level.

6 Alternative Methods of Reasoning

In the most recent phase of our work, we evaluated alternative approaches to providing the required reasoning capabilities. We grouped the alternatives into three categories:

- *Ad hoc* capabilities provided by expert system shells
- Solutions based on standard logic, including such languages as Prolog

- Non-standard logical formalisms such as fuzzy, modal, and temporal logic

These categories overlap, but the classification is useful for comparative purposes. The three categories represent distinct levels of technological and mathematical maturity. The first category, expert system shells, represents the most mature technology and is part of the mainstream of current information processing systems. The second category, which includes logic programming languages and automated theorem-proving systems, is the best understood from the point of view of formal semantics and mathematical properties. The mathematical and semantic foundations of non-standard logic are not as well developed. No concensus exists on the required content of temporal, evidential, and other systems of non-standard reasoning. The practical utility of such formalisms is still unproven (beyond the simple features that have been implemented in some expert-system shells). We regard the third category as worth investigating, however, because it addresses some of the logical requirements more directly than the other categories.

Table 5 summarizes our findings. We see that all three approaches contribute significantly to handling uncertainty, and all three potentially contribute to learning. Abstraction and modeling are supported by the expert system shell features and by standard logic. Temporal reasoning and planning are supported by the expert system shell features and by non-standard logic. Provision of explanations is apparently supported only by the expert system shell features.

A complementary view of the analysis is obtained by considering the specific contributions, actual or potential, of each of the three solution categories. The chief contributions of the expert system shells are 1) object-oriented capabilities for abstraction and modeling, and 2) partial but practical implementations of non-standard logic for handling uncertainty and reasoning about time. The chief contribution of standard logic lies in the flexibility of the logical language for expressing a wide range of concepts (useful in handling uncertainty and in learning). Another potential contribution of standard logic lies in the use of formal methods for knowledge maintenance. The chief contribution of non-standard logic lies in suggesting constructs for handling uncertainty and reasoning about time.

The weakest support appears to be in the areas of planning and providing explanations.

Table 5: Approaches to Meeting Reasoning Requirements

	Expert System Shells	Standard Logic	Non-Standard Logic
Handling uncertainty	Certainty factors Viewpoints or worlds Knowledge maintenance	Disjunction Existential quantification Higher-order logic	Evidential reasoning Possible worlds semantics Modal logic Multiple-valued logic
Reasoning temporally	Viewpoints		Temporal logic Possible world semantics
Abstracting features of a problem or situation	Objects Contexts Inheritance relations	Many-sorted logic	
Planning	Agendas		Possible world semantics Temporal logic
Learning	Knowledge maintenance functions Unstructured knowledge	All combinations of logical connectives Quantification	Full range of non-standard constructs
Modeling other Agents	Objects Contexts Inheritance relations Unstructured facts	All combinations of logical connectives Quantification Higher-order logic	
Providing explanations	Metarules Justifications Deduction rules		

BEST COPY AVAILABLE

In addition, support for learning is principally in the form of knowledge maintenance; support for the creation and refinement of hypotheses, i.e., the problem usually known as "machine learning," is minimal.

Continued research into non-standard logic, knowledge representation, automated theorem proving, and machine learning may help to close some of the gaps. Research and prototyping in non-standard logic will provide additional or improved means of handling uncertainty and reasoning about time. Knowledge representation research will provide methods for abstraction and modeling, and for developing explanation facilities. Automated theorem proving techniques are required for automated planning and knowledge maintenance. Finally, in order to achieve systems capable of hypothesizing and refining solutions to real-world problems, continued research into machine learning techniques is necessary.

References

[1] ISO TC97. *Basic Reference Model.* International Standard ISO/IS 7498. 1984.

[2] Bailin, S., Moore, J., Hilberg, R., and Murphy, E. "A Logical Model of Cooperating Rule-Based Systems in Space Mission Ground Control Facilities." *Twenty-seventh Aerospace Sciences Meeting*. American Institute of Aeronautics and Astronautics. January 1989.

[3] Silverman, B. "FACILITY ADVISOR: A Distributed Expert System Testbed for Spacecraft Ground Facilities." *Proceedings of the Expert Systems in Government Symposium.* IEEE Computer Society. October 1986.

[4] Erman, L., Hayes-Roth, F., Lesser, V., and Reddy, R. "The HEARSAY-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty." *Computing Surveys, vol. 12, no. 2*, pp. 213-253.

[5] Gasser, L. Bagranza, C., and Herman, N. "MACE: A Flexible Testbed for Distributed AI Research." *Distributed Artificial Intelligence*, ed. M. Huhn. Morgan Kaufman Publishers, Inc. Los Altos, CA. 1987.

[6] Durfee, E., Lesser, V. and Corkill, D. "Cooperation Through Communication in a Distributed Problem Solving Network." *Distributed Artificial Intelligence*, ed. M. Huhn. Morgan Kaufman Publishers, Inc. Los Altos, California. 1987.

[7] Green, P. "AF: A Framework for Real-Time Distributed Cooperative Problem Solving." *Distributed Artificial Intelligence*, ed. M. Huhn. Morgan Kaufman Publishers, Inc. Los Altos, California. 1987.

[8] Agha, K. *ACTORS: A Model of Concurrent Computation in Distributed Systems.* MIT Press. Cambridge, MA. 1986.

[9] Smith, R. and Davis,R. "Frameworks for Cooperation in Distributed Problem Solving." *IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-11, no. 1.* January 1981.

BLANK PAGE

Synthetic Organisms and Self-Designing Systems*

W. B. Dress

Instrumentation and Controls Division
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831-6007

Abstract

This paper examines the need for complex, adaptive solutions to certain types of complex problems typified by the Strategic Defense System and NASA's Space Station and Mars Rover. Since natural systems have evolved with capabilities of intelligent behavior in complex, dynamic situations, it is proposed that biological principles be identified and abstracted for application to certain problems now facing industry, defense, and space exploration.

Two classes of artificial neural networks are presented—a non-adaptive network used as a genetically determined "retina," and a frequency-coded network as an adaptive "brain." The role of a specific environment coupled with a system of artificial neural networks having simulated sensors and effectors is seen as an ecosystem. Evolution of synthetic organisms within this ecosystem provides a powerful optimization methodology for creating intelligent systems able to function successfully in any desired environment.

A complex software system involving a simulation of an environment and a program designed to cope with that environment are presented. Reliance on adaptive systems, as found in nature, is only part of the proposed answer, though an essential one. The second part of the proposed method makes use of an additional biological metaphor—that of natural selection—to solve the dynamic optimization problems that every intelligent system eventually faces. A third area of concern in developing an adaptive, intelligent system is that of real-time computing. It is recognized that many of the problems now being explored in this area have their parallels in biological organisms, and many of the performance issues facing artificial neural networks may find resolution in the methodology of real-time computing.

Introduction

The Strategic Defense System is archetypical of a certain class of complex problems that are becoming increasingly important to defense and industry as the 21st century nears. Additional examples of such problems include optimized control of nuclear power plant clusters, design of new and specific molecular medicines, managing the space station, and controlling unmanned planetary exploration vehicles. The complexity of these and similar problems raises serious questions concerning the usual methodology of hardware and software design and makes impossible demands on current methods of reliability testing and system verification. The present approach in treating complex systems is to create a simulation presumed to be representative of the actual system in its essential details. Studying a simulation is thought to be a practical alternative to reality when issues of complexity, prohibitive expense, and impossibility of adequate testing are concerned.

The need for an accurately detailed description of the physical system components and their interactions becomes paramount, as the behavior of the simulation is the basis for developing strategies to cope with real systems. This points to what may be a major flaw in current software simulation and modeling philosophy, as any change requires extensive reprogramming of major parts of the entire simulation. Thus, the predictive power of a simulation to be used for the design of a complex system is easily compromised.

---

*Research performed at Oak Ridge National Laboratory, operated by Martin Marietta Energy Systems, Inc., for the U.S. Department of Energy under Contract No. DE-AC05-84OR21400.

"The submitted manuscript has been authored by a contractor of the U.S. Government under contract No. DE-AC05-84OR21400. Accordingly, the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes."

Incompleteness of information concerning the real, physical systems being simulated imposes another intolerable burden on the simulations and support teams. In addition to the reliance demanded from implementation hardware (sensors, communications, effectors, and processors), we are demanding that programmers perform flawlessly under extreme stress of time constraints and imperfect knowledge. This is clearly unacceptable, as anyone who has ever attempted to write, debug, and run even the simplest program can attest.

This paper results from a search for a methodology to attack these very real issues of hardware and software complexity, reliability, and dynamic variability; and to show how software systems might become self-designing, overcoming both the severe constraints noted above and providing the confidence essential to deployment by ensuring reliable and correct functioning in a changing environment. The ideas presented below are still in their infancy, but they have been partially tested with encouraging results. The main research effort is to determine which principles to abstract from nature, the extent of abstraction necessary, and the details required for creating intelligent machines; for it is only through adaptive, intelligent systems that the limitations noted above can be overcome.

The problems of modeling and simulation are discussed first, and a new principle of software engineering is proposed. The question of whether a complex system can be simulated is raised. Examination of issues leads to a proposal for creating synthetic organisms to solve certain complex problems. Two network models are presented as a vehicle for implementing a self-designing, complex system. Results of the two evolutionary programs based on these networks are discussed, and a number of possible extensions to the methods are given.

Simulation & Modeling

For problems of sufficient complexity, a step-by-step simulation is the most efficient means of obtaining predictions of system behavior. Wolfram[1] has argued that the behavior of certain systems may be effectively found only by an explicit, step-by-step simulation, and he considers such systems to be "computationally irreducible." Wolfram's argument amounts to showing a contradiction between the assumptions of a universal computer for such calculations and the existence of an algorithmic shortcut for the simulation. Physics and engineering are concerned primarily with the computationally reducible, while most biological systems are computationally irreducible. For example, "the development of an organism from its genetic code" may well belong to the latter class.[1] Wolfram goes on to suggest that "the only way to find out the overall characteristics of the organism from its genetic code may be to grow it explicitly. This would make large-scale computer-aided design of biological organisms, or 'biological engineering,' effectively impossible: only explicit search methods analogous to Darwinian evolution could be used." [1]

Given the dynamic nature of a complex, real-time control problem, the phrase "biological organism" may be replaced with "software" and "biological engineering" with "software engineering," extending the range of applicability of the previous sentence. Wolfram's suggestion then becomes a new principle of software engineering for truly complex problems. It is this principle that we wish to explore along with neural networks and adaptive systems.

Complex Systems: Where Simulations Fail

Why are we concerned with biology and problems of computational irreducibility? Artificial neural networks are well-understood computational structures firmly rooted in the mathematics of systems of first-order differential equations, and their proponents claim that many pressing problems will yield to the new paradigm of computational neuroscience. On the other hand, a biological system is one that lives in and has been optimized for a certain dynamic ecosystem. Such systems are complex and not well understood from the simple-system perspective. Evidence is accumulating from many quarters that systems combining information management and real-time control of complicated hardware are likewise not simple. By the very nature of an algorithm, algorithmic methodologies developed to cope with simple systems will most assuredly fail when applied to these complex problems. Indeed, it is already evident that expert systems (deterministic decision trees) become brittle when the application domain is slightly altered, as does any algorithmic structure when used outside its range of applicability. Note that the *ad hoc* addition of "fuzzy reasoning" by adding Bayesian logic or fuzzy sets does not cure this problem: once the ranges of variation are specified, the system is still essentially deterministic. Although simulation may well be a practical way to study certain problems, it is too much to hope that the limitations imposed by brittle programs and inadequate knowledge can be overcome by a purposefully designed simulation.

To go beyond simulation to a synthetic organism that solves a complex, real-time problem in an effective manner is seen as the next logical development in computer science. A simple system, by definition, is one allowing separation between states and dynamical laws, i.e., the intrinsic nature of the system and its response to external effects.[2] The author of this view of physical systems, Robert Rosen, suggests that "any system for which such a description cannot be provided ... [is] *complex*. Thus, in a complex system, the causal categories become intertwined, in such a way that no dualistic language of states plus dynamical laws can completely describe it."[2] Computability of complex systems may be examined by considering the well-known Church-Turing Thesis, which asserts, in essence, that any material process can be simulated. That is, the difference between actual points in a state space of a real system and corresponding calculated points in the space of the simulated system can be made arbitrarily small by sufficient calculational effort.

There are two reasons—one practical, the other fundamental—why any actual simulation will fail when asked to perform beyond strict design limits. The practical reason has been discussed above as due to imperfect knowledge of the reality being simulated; the second, more fundamental reason, is based on Rosen's discussion of complex systems. Gödel[3] has shown that the Church-Turing Thesis fails for arithmetic. Thus, for example, it is not possible to encode the whole of arithmetic into input strings for a Turing machine in such a way that every truth of arithmetic is provable as a theorem. Rosen uses this fact as a point of departure to discuss differential equations as universal simulators.[4] He then goes on to show that "general vector field[s] cannot be described to a Turing machine, ... [and] since they cannot be encoded, they cannot be simulated. It is precisely here that Church's Thesis fails in analysis. In a precise sense, most orbits of such a general vector field are not computable." [4]

Rosen seems to be suggesting that since algorithms (computer programs) can indeed compute any computable function, and since behaviors of certain complex systems are not computable by not possessing a complete syntactic description, there can be "no independent, inherent distinction between hardware and software"[4] as the Turing machine demands. Simulations can at best repeat what "organisms *have already done*; not the things they will do."[4] A real, parallel, asynchronous neural network model is therefore necessary to emulate non-computable functions—the orbits of the general vector field. Thus, we must progress from the neural network simulations of today to actual neural network systems of sufficient generality and power to mirror real neural activity at some level of abstraction such that they become actual synthetic organisms that learn to cope with the problems we need to solve. Only then will we have achieved our goal of creating machines with enough intelligence (or any intelligence at all, for that matter) to cope adequately with the types of problems considered here.

A Synthetic Intelligent System

To create a system exhibiting the ability to deal successfully with a complex and changing environment, a biological metaphor of an ecosystem inhabited by potentially intelligent agents is employed. The ecosystem may be changing on a continual and slow basis, as all natural systems do. A group of similar organisms presently adapted to the ecosystem is considered to be a species. It is this species that adapts to the changing environment on a genetic time scale when changes are outside a certain optimality range for the present members of that species. The individual members of the species adapt to changing conditions on a time scale determined by plasticity of the organism; this plastic period may last for the lifetime of the individual or merely during an infant and juvenile period. The important distinction is that the genetic time scale for change is much longer than the individual time scale. If changes occur too rapidly for any given individual to adapt, but not so severely as to be out of range of the available genetic pool, then a given individual may fail, but the species as a whole will adapt. If changes occur too rapidly over too extreme a range, the species becomes extinct.

Reliance on a single program or set of programs will eventually prove fatal (even if completely error free), whereas a (possibly large) set of (possibly virtual) programs can form a genetic pool, allowing mutations and crossover to bring about the evolution of a successful program. This, then, is the thrust of this work: to set up conditions in which an intelligent system may create itself through evolution.

Role of the Environment

The approved software-engineering procedure for writing a program is to start with a set of specifications that describe the desired results. Another way of viewing this process is to suppose we are constructing a function (the algorithm) that maps from a subset of internal machine states, indicated by the statements of the program, onto a subset of the possible actions that a machine can effect in the external world. The subset of this range of actions is precisely those results specified in the top-level design stages (if all has gone correctly). The programmer's job is to select the most appropriate subset in the domain of the mapping (the set of all possible machine states or statements in the programming language) that best map onto the range.

The method proposed here turns this process around and dynamically closes the loop between high-level specifications and program statements. It is this closure that is usually neglected once the initial design specifications have been made. This neglect is ultimately responsible for the brittle software systems that we are so reluctant to allow to control complex machinery. Even a conscientious effort to close this loop will not solve the problem for those systems required to function in open environments—the loop needs continuous and dynamic closure even as the environment changes.

If the domain of the function (algorithm, program) is widened to a much larger virtual space of possible mappings, the task then becomes one of evaluating the behavior of a given program instance in its range. Evaluation is generally a much simpler algorithm and may be thought of as the inverse mapping from the range of possible actions onto the domain of virtual programs. Of course, a sufficiently generalized representative of the virtual program space must be created initially for this method to work. How this might be accomplished is discussed below.

The environment is an essential part of the ecosystem we wish to control. A specific ecosystem may consist of sensors, databases, computing engines, available software libraries, space platforms, offensive and defensive weaponry, the immune system and invading organisms. The group of functional organisms in an ecosystem act upon and react to the environment; they are in fact inseparable from the ecosystem, which is a nonlinear dynamic system of interacting parts.

The programmer/designer becomes a policy maker by providing the system with a "fitness" function that evaluates success in the environment and thus closes a loop that is normally recognized only at the system specification level in current software engineering practice. All interaction between the designer and the system is through this high-level policy algorithm, which monitors overall behavior, guiding the system to a region of local optimal functionality. Set too tight a specification, and the ensuing system loses adaptability that may be essential at some future time (e.g., when an unexpected terrain is encountered by an exploratory vehicle). Too loose a specification, and the ensuing system will behave other than desired, and may fail by default. The key point is that the closure between function and specification in a self-designing system is a continuous process.

The question of reliability assurances in the form of proofs of correctness will surely arise in the course of presenting this new (but very old) paradigm. For intelligent systems, such proofs are not only impossible, they are not even applicable: can a proof of correctness be found for the President of the United States? How, then, can we prove that a given intelligent being is going to perform correctly? The answer is that the question is ill-conceived; it is a question borrowed from one domain and forced onto another. The correct question is: Can we reasonably assume that the job will be done correctly by a certain individual? The only conceivable answer is one involving estimates and limits based on the experience of the evaluator and the candidate. The main point is that when relying on provably correct algorithms for complex, real-world situations, failure is inevitable because, sooner or later, the environment will change in an unexpected manner. With an adaptive, intelligent system, a provably correct answer may never become available in spite of our best computer science departments. On the other hand, total failure is not inevitable—the adaptive, intelligent system will quite probably muddle through to victory one way or another. Thus there is a kind of complementarity here—a too-restrictive policy measure that guarantees the existence of a correctness proof will result in brittle programs, while a more liberal policy will deny a such a proof but allow intelligent and adaptive programs to evolve.

An Optimized Retina

If we consider a retina to be a filter for complex spatial patterns that extracts certain types of information (whether in the visible spectrum or not is immaterial), then a generalized retina is a necessary part of any entity required to function in an environment that possess objects crucial to the entity's success. Pattern recognition is only one of the functions such a device must possess. Thus we look to the role of a retina as fundamental to machine perception.

Again, looking to natural systems for guidance, a retina may be specifically optimized to recognize certain features. Frog retinas responding strongly to nearby moving insects, migrating birds orienting their routes via constellations, and babies responding to abstract human faces all come to mind as genetically designed recognition systems. Hubel[5] has shown that the human retina is also well designed for sensitivity to edges, orientation, and motion in the field of view. But such generality may not be necessary in certain applications such as identification of specific objects in a restricted environment.

A retina was constructed from a neural network based on early work in pattern recognition by Bledsoe and Browning[6] and a later elaboration by Uhr.[7] The standard n-tuple algorithm[6,7] was recast as a feed-forward neural network consisting of randomly connected feature detectors. Each feature detector has n inputs from n different retina cells (the simulated photoreceptors). In Figure 1, n is chosen to be 3, so there are 2^3, or eight, outputs of the feature detector, each of which may correspond to a feature of one or more categories that are learned by the network during a training phase. Both learning and recall involve direct access from input cells to feature detectors to the summing category nodes—no expensive relaxation process to minimum energy states[8] is necessary, nor is backpropagation[9] of errors during the learning process required—there are no graded errors in a Boolean system. Due to the statistical nature of the connectivity and the requirement that reasonable samples of each category be presented during training, the network is both fast and reasonably immune to noise—two desirable features for real-time, real-world applications.

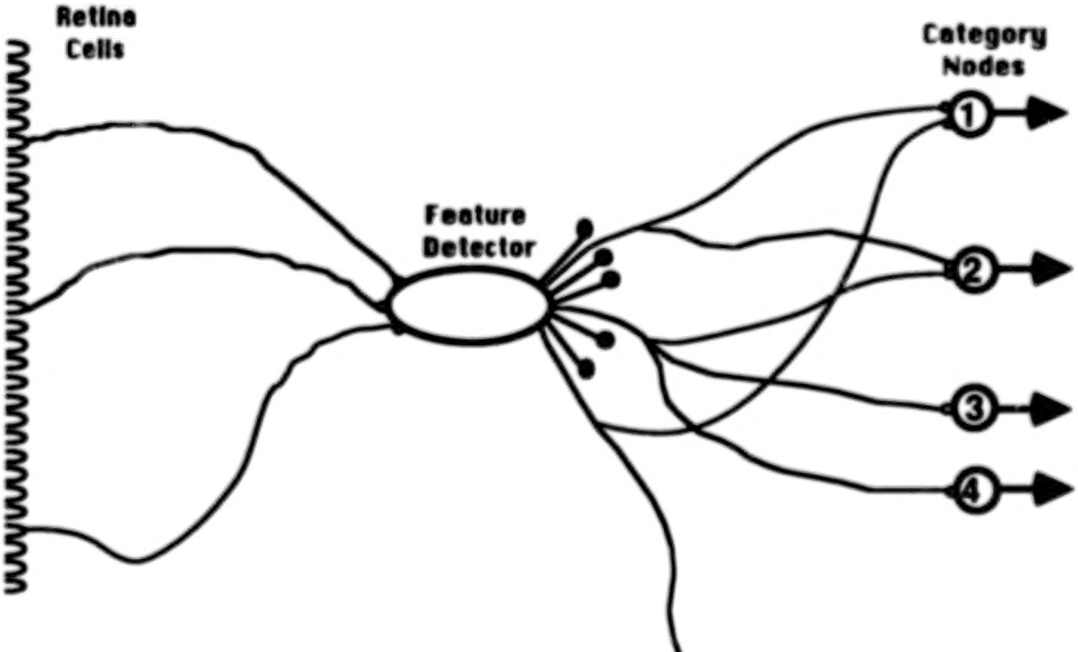


Figure 1. The feature-detector retina model is a feed-forward network activating category nodes. Activity in the retina cells is grouped into features by the m feature detectors, where m = N/n, N is the number of retina cells, and n is the order of the n-tuple. Each feature detector has a maximum of 2^n output lines connected to category nodes during training. These lines activate category nodes during recognition. Category nodes are added to the network as needed.

Both the connectivity of the network and the contents of the memory may be taken as genetic specifications. In an experiment described below, only the memory cells are subject to mutation. An alternate approach involves an evolution of feature detectors for particular sets of patterns by mutation of the connectivity between the input cells and the feature-detector nodes. In this way, invariants of the set of patterns will be encoded into the connectivity of the network. The adaptability of this type of network takes place on the evolutionary time scale—much longer than the plasticity time constants of the adaptive brain to be considered next.

An Adaptive Brain

While we can conceive of a brain without its emergent property of intelligence (indeed, examples abound), the converse of intelligence without a central nervous system (CNS) is much more difficult to imagine. The CNS used in the present work follows closely a model originally developed by Browning[10] for the Sandia Corporation. Browning chose a system modeling those biological neural networks that make use of frequency encoding of information transmitted by nerve impulses.[11] It is unclear whether frequency-coded information flow in the brain is fundamental to brain operation or whether it is merely a convenient solution to the problem of communication in a noisy environment between low-reliability components. However, recent work indicates that information is coded in the actual axon pulses and is indeed an important means of information processing, at least in certain areas of the brain.[12] Approximate coincidence of information packets traversing the network is a stringent requirement imposed by a frequency-coded network and may underlie the discrimination capabilities of the CNS. The degree of abstraction allowable in a simulated CNS is an open question—can we talk about frequency as a function of time as done by many researchers[13] or must the actual axonal spikes be simulated individually? The present work does not make the simplifying assumption of an average, differentiable frequency function, $v(t)$, for the neuron's output; instead, it simulates each axonal spike separately.

The model, as implemented, consists of a few hundred frequency-responsive neurons or nodes. Each node has an arbitrarily chosen number of inputs from other neurons and, on the average, a like number of outputs simulating the distribution of axonal spikes. The average connectivity is predominantly from a row of input nodes through several rows of internodes, which are not necessarily "hidden,"[14] to a row of output nodes. There is a high probability of connections in the forward direction (input to output) and a low probability in the lateral and reverse directions. The distribution function is a Rayleigh function modified by an elliptical angular distribution with the major axis aligned along the forward-backward direction.

"Synapses" are formed at the junction of the input of one node to the output of another and are modeled by a pointer associated with an input node that refers to a memory location associated with an output node. The synaptic efficacy of transmitting an axonal spike through a junction is analogous to the "weight" found in many artificial neural network models.[14] This weight is modifiable, and the modification algorithm may be altered to investigate various theories of learning and memory. To date, a simple Hebbian algorithm has been used, as well as a frequency-based version of the BCM synaptic modification.[15] Other learning theories under investigation are the drive-reinforcement model,[16] and the dual-synaptic population model.[17] Detailed comparisons of these various models of learning are not available at this time, although each of the algorithms produces reasonably satisfactory results in that the system of neurons and synapses undergoes self-organization related to the environment imposed.

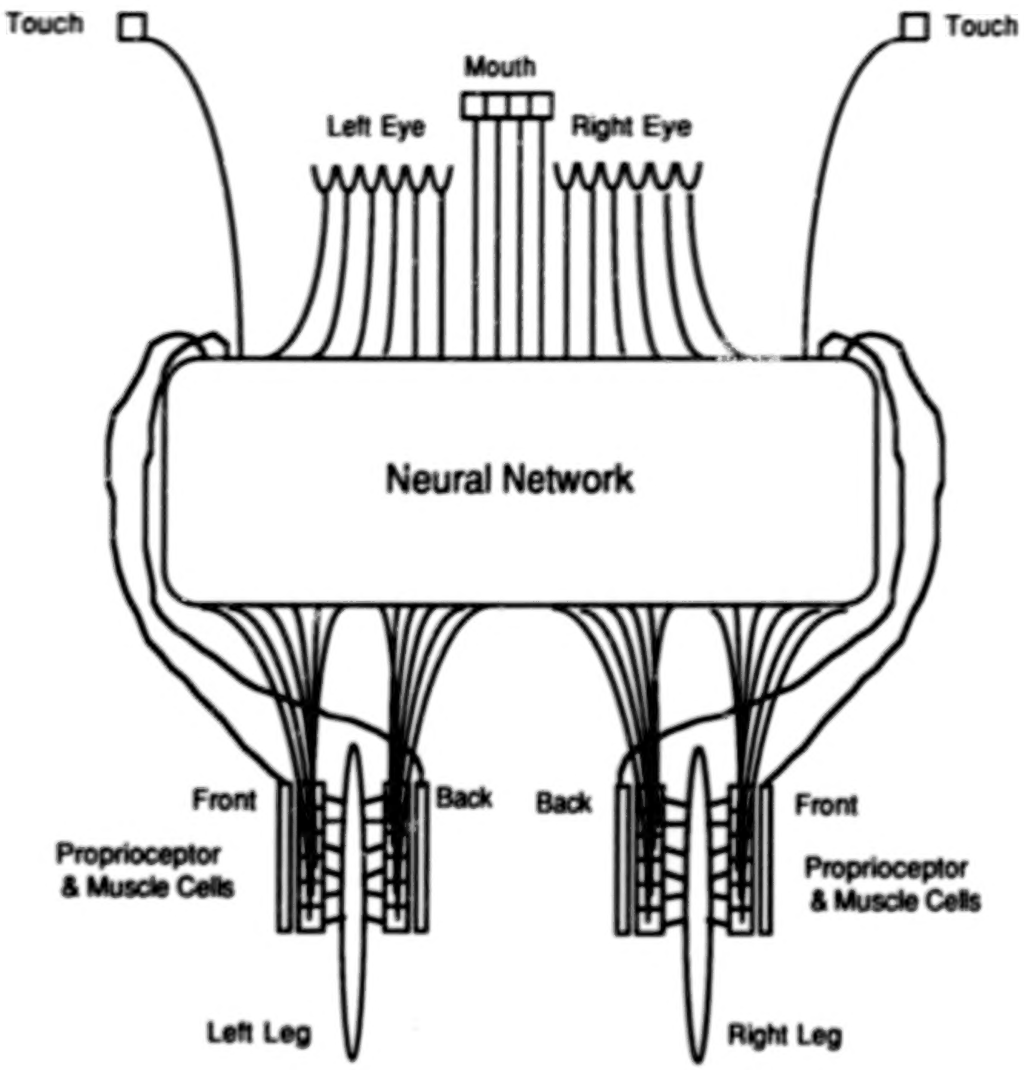


Figure 2. A synthetic organism is constructed from an adaptive, frequency-coded neural network having sensors and effectors for interacting with the environment. The organism is presented here as an insect, but the paradigm is not limited to a particular class of phenotypes.

A means of interacting with the environment was added in the form of simulated sensors (vision, taste, and touch) and effectors (groups of muscle cells). The resulting synthetic organism is shown in Figure 2. Depending on the sensors and effectors given such an organism and the environment in which it is required to function, the designer may demand anything from an artificial rat for classroom experiments in animal psychology to an autonomous vehicle required to explore the Martian surface. The biological basis of synthetic intelligence is versatile enough to produce a wide variety of successful "species." These extensions and variations are the goal of future research into electronic (and eventually, electro-mechanical) life forms.

Evolution and Self-Design

In a situation of sufficient complexity (as discussed above) where proofs of correctness are unattainable, the construction of infallible systems is impossible without an omniscient programmer. Darwin[18] has given us a model for the creation of optimal systems in artificial universes (independent of its correctness in the real universe). The omniscient programmer, even if possible, is no longer needed for the creation of complex hardware and software systems when the principles of evolution are employed—a fallible program can improve its own behavior. Thus, we are on the verge of establishing the necessary conditions whereby electronic life can arise and evolve in the computer, and optimize its behavior in environments of our choosing guided by a policy of our choice. This is an extremely powerful paradigm for the design of systems to handle complex, biological-like problems, and it will lead to a revolution in the science of complex systems. Over the years, a few individuals have become interested in these ideas. One of the early investigators was W. W. Bledsoe[19], who examined some of the possibilities and problems associated with genetic models in computer science. Fedanzo[20] gave a more recent admonition to follow Darwinian precepts in problems concerning data base optimization.

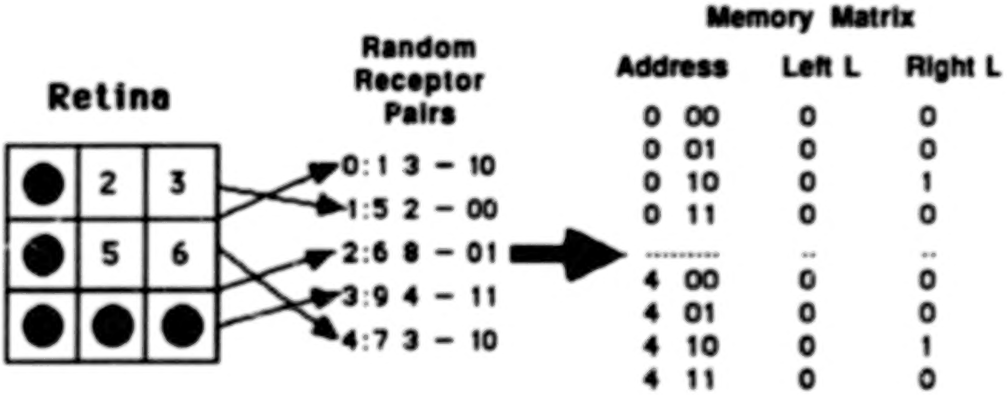


Figure 3. A classical n-tuple pattern matcher is shown for n = 2 and a 3 X 3 input array. A typical pattern is shown in the array on the left. The table in the middle shows a possible arrangement of the five possible (mostly) exclusive, randomly chosen pairs of retina cells, and the formation of subaddresses is indicated. The table on the right shows the memory matrix necessary to store two categories, one per column. The pair labeled 0 consists of the ordered retina cell pair 1,3. Cell 1 has a pixel turned on, cell 3 does not; the corresponding subaddress is therefore 10 (binary) or 2. Thus the memory matrix has an entry (if trained) at address 0, subaddress 2 in the right column, corresponding to the right-facing L-pattern.

The usual approach to adaptive systems can be made into a Darwinian approach to self-designing systems by carefully separating design and performance specifications from adaptive structures (synaptic weights, polynomial coefficients, etc.) belonging to the individual organism or, in this case, the executing computer program. One of the most noticeable differences between an adaptive genome and an adaptive system concerns time scales: adaptation in the usual sense occurs on a short time scale and within a certain program that is self-organizing in response to the environment or problem. In the case of genome evolution, the time scale is much longer, extending over many individual organisms (programs) interacting with their environment and resulting in the self-organization of the genome itself.[21]

Positive and Negative Selection

There are a wide variety of selection strategies to choose from when considering optimization based on Darwinian principles. The main idea is to introduce variations and demand that reproductive success depend on fitness (in Darwinian theory, the two concepts are synonymous). Here, we consider the effects of the two general stragegies of positive and negative selection on fitness. Both of these strategies are amenable to simulation in a relatively simple system. The principles discussed below are well known to evolutionary biologists (e.g., Mayr[22] and Kimura[23]), and to some animal breeders, but seem relatively unknown to computer scientists.

Browning[24] suggested a simple experiment done with a data-structure version of the pattern-recognition system described above. The pattern memory is a set of binary cells addressed by patterns in the retina (see Figure 3). Mutations are made by logically complementing cell contents. The algorithm followed involves a mutation in a single, arbitrarily chosen memory cell and measuring of the success of the retina in recognizing the set of L's, both normal and reflected in the vertical. Inspection of the figure shows that there are 9 such L's possible in each orientation on a 3 X 3 grid (we are not considering reflections about the horizontal). The scores of each of the 18 L's are computed by counting 1 for each cell addressed by a particular L pattern (the L shown in Figure 3 would score 0 for the "Left" category and 2 for the "Right" category for a total score of 2). The score then becomes the "fitness" of that particular mutation.

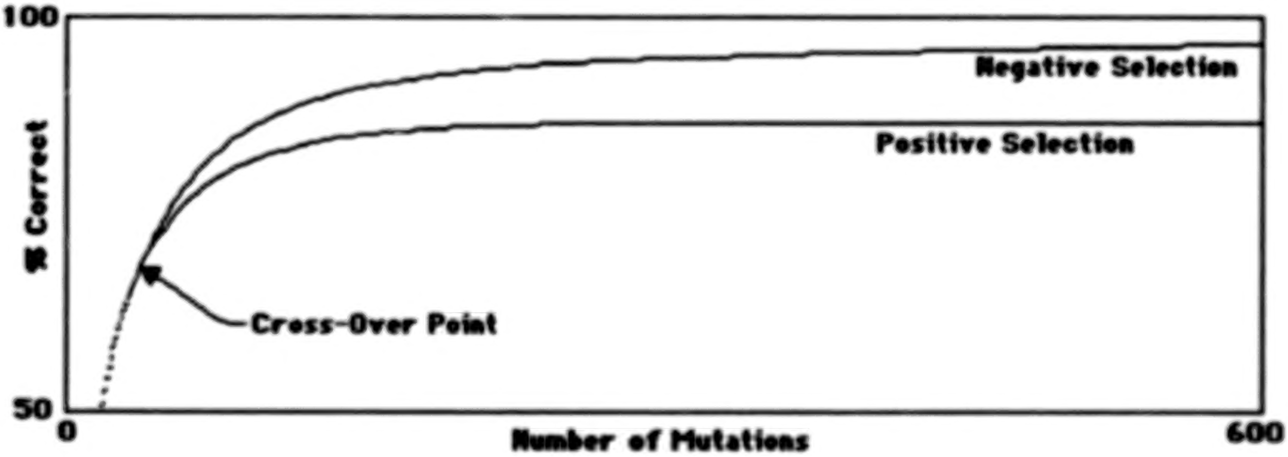


Figure 4. Results of a simple artificial genetic experiment on the memory matrix shown in Figure 2. In the 20 memory cells, 600 mutations were made for each selection strategy (see text). The strategy corresponding to negative selection clearly outperforms the positive selection case.

The experiment was run for two different selection strategies, positive and negative. "Positive" is defined as: Accept a mutation if it produced a higher score, otherwise reject it. "Negative" is defined as: Reject a mutation only if it produced a lower score. Thus we are selecting against failure, *not* for success. As Figure 4 shows, the difference is dramatic. Positive selection starts with a faster slope initially, but saturates quite early (below about 250 mutations for this 20-cell system). Negative selection, however, quickly overtakes the positive, and is still showing improvement at the 600-mutation point.

Sex and Genetic Algorithms

Employing negative selection results in improvements in the optimization algorithm chosen. Additional acceleration of the evolutionary process is possible in a sexual species where there is an opportunity for individuals to pool complementary portions of genetic material as shown by Ulam and Schrandt.[25] Application of the genetic algorithms as pioneered by Holland,[26] especially the use of cross-over methods,[27] has been shown to result in accelerated optimization for these classifier systems. There is every reason to suppose that variants of these methods, involving a careful separation between the "phenotype" and the "genotype," would dramatically accelerate the process if applied to a collection of individuals forming a gene pool.

Results & Future Directions

One aspect of intelligent behavior is the ability to solve a problem in a surprising fashion. The system described above has generated such a surprising solution to a presumably simple problem. The problem posed to the system was to optimize the behavior of the synthetic organism by avoiding the boundaries of the environment yet keeping in motion to explore and interact with that environment. A simple fitness function set the policy by evaluating each time step of the synchronous system. The system was left to evolve on its own. The expectation was that the various parameters determining the tactile sensitivity would develop to the point where touching the boundary would initiate a sequence of network node firings effectively demanding retreat of the organism from the walls. Since the high frequency felt at the boundaries naturally proves disruptive to a frequency-coded network, as shown by previously,[28] it was natural to assume that this inherent capability would be optimized. The surprise was that this did not happen. Instead, the synthetic creature evolved—after more than four hundred generations—into a new species whose locomotion was predominantly backward. The new organisms walked backward in a very efficient manner, occasionally turning around to sense objects in the environment. There was no touch sensor on the rear, so posterior collisions with the walls had no effect on the fitness function and could not disrupt the activity of the network.

Thus, a group of parameters, or a "gene" of the system, altered to solve the problem in an efficient, surprising, and biologically reasonable manner. Indeed, one of the induced mutations discovered in *C. Elegans* (a microscopic, 850-celled worm having approximately 300 neurons) causes this very behavior.[29] Several *unc* (for uncoordinated) mutations affect the ability of the worm to move forward and backward. In particular, *unc*-4 prohibits backward motion entirely, while the *unc-7* mutation causes predominantly backward motion.[30]

As an exercise in understanding a complex system, a preliminary analysis was made of the parameter file (defining the structure and behavior of the simulated organism) before and after the evolutionary episode. The new species developed somewhere prior to 456 generations involving 1537 mutated individuals. It was assumed that a parameter residing in a group responsible for the dynamics of the muscle motion would be identifiable as responsible for the reverse locomotion (indeed, there is a parameter specifying the degree of asymmetry in the extensor-to-flexor contractions). This hope was dashed when the standard deviation of the percentage change of the parameters in the muscle-dynamics group was found to be not significantly different from that of any other of the functional groups. Indeed, the asymmetry parameter changed in the direction of increased forward impetus by about 10% rather than in the direction of reverse locomotion. The number of random trials necessary to alter the asymmetry parameter sufficiently to cause predominantly backward motion is approximately 100^5—much larger than the <1537 trials actually needed. Thus, the selectionist method of optimization is far more effective than a random method would be.

We have given an example of a system whose behavior is clearly known, but whose internal causes are not yet understood. The situation is analogous to one's pet dog: you can never understand an organism as complex as a dog, but you sure can make it sit whenever you wish (almost). Dogs have proven their reliability in complex, difficult, and demanding situations throughout history, yet they are neither understandable (in the reductionist sense) nor provably correct.

The Future

Access to faster processors operating in parallel configurations will allow a number of additional techniques, all taken from biology, to be applied to the creation of self-designing systems. Sex, in particular, was mentioned above. Other means of accelerating evolution involve interaction between individuals of the same or different species. A process of coevolution, or "arms race" as in a mutual selection for speed in cheetahs and their prey, gazelles, is one example. Here, selection pressures force one, then the other species to excel marginally. The result is either the extinction of both or two very fast animals. Similarly, direct competition for a particular resource, such as the food objects in the simulation described in this paper, would certainly accelerate the optimization process.

All of these methods require very fast hardware and sophisticated simulation languages (for specifying ecosystems as well as neural networks). An ideal would be to let synthetic organisms interact and compete simultaneously on a set of parallel processors. Policy for coevolution between a defensive system and an offensive system would be set for victory of one "species" rather than mutual survival as in the cheetah case. Thus, an immune-system molecule could be synthetically evolved to specifically label a particular virus fragment (both in silico and in vitro).

It is anticipated that the most valuable result of this work will be a system that, upon sensing failure, will enter a mode of accelerated evolution, producing success by replaying and adapting to events that lead to the failure. This would be the ultimate adaptive system. Obvious applications are in a strategic defense system, a survey robot for nuclear power plants, and a method of responding with specific vaccines to the high rate of mutation of the AIDS virus. For NASA, there are likewise obvious applications: a planetary exploration vehicle will probably meet with failure due to unanticipated environmental effects. Any means of turning such an eventual failure into success should be welcome.

Acknowledgments

This work depended on support provided partly by the Instrumentation and Controls Division, partly by the Energy Division, both of Oak Ridge National Laboratory, and recently by the Materials Laboratory at Wright-Patterson Air Force Base.

The inspiration for this work came primarily from Dr. Iben Browning, who has been a consistent and dedicated source of ideas and perceptive and thought-provoking comments since 1986. John Peers of Novix, Inc. has provided invaluable moral and valuable technical support, as well as being instrumental in realizing a super microprocessor created by those geniuses of software and silicon, C. H. Moore and R. W. Murphy, who have my profound gratitude. Without this device, the evolutionary development described above would have been totally impractical.

References

1. Stephen Wolfram, "Complex Systems Theory," in *Emerging Syntheses in Science*, David Pines, ed., Addison-Wesley, Redwood City,1 988.
2. Robert Rosen, "Some epistemological issues in physics and biology," Cpt 22 in *Quantum Implications*, B. J. Hiley and F. David Peat, eds., Routledge & Kegan Paul, New York,1987.
3. K. Gödel, "Über formal unentscheidbare Sätze der Principia Mathematica und verwanter Systeme I," *Monatsheft für Mathematik und Physik*, **38**, 173-98, 1931.
4. Robert Rosen, "On the Scope of Syntactics in Mathematics and Science: The Machine Metaphor," in *Real Brains, Artificial Minds*, eds. John L. Casti and Anders Karlqvist, Elsivier Science Pub. Co., New York, 1986.
5. David H. Hubel, *Eye, Brain, and Vision*, Scientific American Library Series #22, W. H. Freeman and Company, New York, 1988.
6. W. W. Bledsoe and I. Browning, "Pattern Recognition and Reading by Machine," *1959 Proc. Eastern Joint Computer Conf.*, 225-32, 1959.

7. Leonard Uhr, Chapter 3, *Pattern Recognition, Learning, and Thought*, Prentice-Hall, Englewood Cliffs, 1973.
8. John J. Hopfield and David W. Tank, "Computing with Neural Circuits," *Science*, 625-33, 8 August, 1986.
9. P. J. Werbos, "Beyond Regression: New Tools for Prediction and Analysis in the Behavorial Sciences," *Thesis*, Harvard University, 1974.
10. Iben Browning, "A Self-Organizing System Called 'Mickey Mouse,'" Panoramic Research Report, Palo Alto, 1964.
11. John C. Eccles, *The Physiology of Nerve Cells*, The Johns Hopkins Paperbacks Ed., Baltimore, 1968.
12. Lance M. Optican and Barry J. Richmond, "Temporal Encoding of Two-Dimensional Patterns by Single Units in Primate Inferior Temporal Cortex. III. Information Theoretic Analysis," *J. Neurophysiology*, **57** (1), 162- 78, 1987. (See also parts I and II in the same issue.)
13. Robert J. Baron, *The Cerebral Computer*, Lawrence Elbaum Associates, Hillsdale, NJ, 1987.
14. J. L. McClelland and D. E. Rumelhart, Chapter 17, "A Distributed Model of Human Learning and Memory," *Parallel Distributed Processing*, James L. McClelland, *et al.*, ed., The MIT Press, Cambridge, 1986.
15. M. F. Bear, L. N Cooper, and F. F. Ebner, "A Physiological Basis for a Theory of Synapse Modification," *Science*, **237**, 42-48, 1987.
16. A. Harry Klopf, "A neuronal model of classical conditioning," *Psychobiology*, **16** (2), 85-125, 1988.
17. Gerald M. Edleman, *Neural Darwinism*, Basic Books, New York, 1987.
18. Charles Darwin, *On The Origin Of Species*, Facsimile of the First Edition of 1859, Harvard University Press, Cambridge, 1966.
19. W. W. Bledsoe, "The Use of Biological Concepts in the Analytical Study of Systems," Panoramic Research Report, Palo Alto, 1961.
20. A. J. Fedanzo, Jr., "Darwinian Evolution as a Paradigm for AI Research," *SIGART Newsletter*, **97**, 22-23, 1986.
21. W. B. Dress and J. R. Knisley, "A Darwinian Approach to Artificial Neural Systems," *Proc. 1987 IEEE Internat. Conf. on Systems, Man, and Cybernetics*, 572-77, 1987.
22. Ernst Mayr, "The Unity of the Genotype," 71-72, in *Genes, Organisms, Populations: Controversies over the Units of Selection*, Robert N. Brandon and Richard M. Burian, eds., The MIT Press, Cambridge, 1986.
23. Motoo Kimura, *The Neutral Theory of Molecular Evolution*, Cambridge University Press, Cambridge, 1985.
24. Iben Browning, personal communication, 1987.
25. S. Ulam and R. Schrandt, "Some Elementary Attempts at Numerical Modeling of Problems Concerning Rates of Evolutionary Processes," *Physica* **22D**, 4-12, 1986.
26. J. H. Holland, "A Mathematical Framework for Studying Learning in Classifier Systems," *Physica* **22D**, 307-17, 1986.
27. David E. Goldberg, *Genetic Algorithms*, Addison-Welsey, Reading, MA, 1989.
28. W. B. Dress, "Frequency-Coded Artificial Neural Networks: An Approach to Self-Organizing Systems," *Proc. IEEE First Annual Internat. Conf. on Neural Networks*, Vol II, 47 - 54, 1987.
29. Tom Coohill, Western Kentucky University, personal communication, 1988.
30. Jonathan Hodgkin, *Gene List for the Book of the Worm*, MRC Laboratory of Molecular Biology, Cambridge, England, 1988.

BLANK PAGE

INTEGRATION OF PERCEPTION AND REASONING IN FAST NEURAL MODULES

David G. Fritz
Research Scientist
Institute for Artificial Intelligence
The George Washington University
and
Cognitive Information Systems Co.
Silver Spring, Maryland

Abstract. *Artificial neural systems promise to integrate symbolic and sub-symbolic processing to achieve real time control of physical systems. Two potential alternatives exist. In one, neural nets can be used to front-end expert systems. The expert systems, in turn, are developed with varying degrees of parallelism, including their implementation in neural nets. In the other, rule-based reasoning and sensor data can be integrated within a single hybrid neural system. The hybrid system reacts as a unit to provide decisions (problem solutions) based on the simultaneous evaluation of data and rules.*

This paper discusses a model hybrid system based on the fuzzy cognitive map (FCM). The operation of the model is illustrated with the control of a hypothetical satellite that intelligently alters its attitude in space in response to an intersecting micrometeorite shower.

Concept. Artificial perception, cognition, and learning are increasingly possible by imitating natural information processing mechanisms. Information processing in living systems occurs in two major forms, genetic evolution and chemical/ electric cell-to-cell communication. Both of these natural processes are being exploited for what they can contribute to artificial computation. The field of Computational Genetic Algorithms has borrowed ideas from natural evolutionary theory to develop learning and problem-solving programs. Artificial Neural Systems (ANS) have taken inspiration from natural nervous systems in order to model the parallel, distributed processing of the brain. These two currents of thought derive their power from the inherent parallelism of natural solutions to information processing.

Small, low cost information processors were first developed by natural living organisms. This tactic served developing life forms well for hundreds of millions of years. Only late in the evolutionary process did bundles of these small localized neural processors coalesce into large brains to which other peripheral knots of neurons could report in turn. This process of natural develpment can be emulated in artificial systems by initially producing small, intelligent information integrating devices able to categorize and classify local information, and which communicate and coordinate their state with a non-local decision-making center. Such devices would be capable of learning and reasoning, and of operating in continuous real-time.

Given the current investment in expert systems, and the relative immaturity of neural connectionist systems and their learning interfaces, practical applications in intelligent processing enhancement will probably consist initially of expert system/ neural net combinations, rather than of neural nets alone. There are two likely approaches to this near term scenario: symbolic expert systems with neural nets at the data collection points (front ending), and computational systems of mixed representation that allow the close integration of concepts and rules with low-level data. Both of these approaches have relative strengths and weaknesses, but only the second is close to being "natural" in the sense discussed here.

The first of the two approaches consists of the straightforward combination of an expert system with whatever neural net models are needed to provide input at the data collection points of the expert system's rule graph. In its simplest form, the integrated program combines a normative expert system with selected ANS. Whereas the expert system typically queries a user or a data-base/ knowledge-base for information, the integrated program also queries, or extracts information from, neural nets. In this system, machine learning occurs at the sub-symbolic level in the neural nets. However, neural net input-output patterns can also be extracted as symbolic rules for incorporation into the expert sytem's rule graph as they are learned.

The mixed representation approach provides for systems that allow the close integration of high-level concepts with low-level data. These systems do both the data collation and a degree of symbolic level processing as a unit. The system thereby behaves as a symbolic/ sub-symbolic hybrid. The hybrid is different than the first approach described above in that a major portion of the data hybrid is implemented entirely under the ANS paradigm. The hybrid is different than a connectionist expert system in that the neural model does not simply replicate the functionality of an expert system, but is aimed at fusing in real-time the information provided by sensors and through conceptual relationships.

On this basis, the hybrid system's strength consists of the capability to provide a fine-grained integration of symbolic concepts with sub-symbolic information. Operations on the two types of knowledge occur at the lowest computational level. Incomplete, innacurate or contradictory rules are buttressed by the natural fault tolerance/ graceful degradation of the neural elements, providing for automatic truth-maintenance support. In addition, the automatic translation of sub-symbolic representations into symbolic rules can occur in the same computational neighborhood, such that the high level conceptual portion of the system learns as well and as easily as the neural elements do.

Both of these approaches are useful in their own right. But, whatever approach is chosen to enhance intelligent processing by way of neural models, it is important to address the art and practice of neuralism as well as the science. Science provides testable ideas but does no work. Useful work will result from realized improvements in practice by way of new ideas. Neural models provide two basic beneficial improvements: conceptual and associative learning from sub-symbolic information, and simultaneous processing activity. The first of these will be important over the long run in addressing the knowledge acquisition and maintenance bottleneck. The second is of more immediate relevance.

The standout value of connectionist systems is going to be in harnessing parallel behavior. It is important, therefore, to look to eventual hardware implementations of chosen neural models in order to provide this parallelism. Without a hardware realization, the often-used signal Hebb law is basically another algorithm, of the many excellent ones available. New algorithms may or may not be better than those that came before, depending on the problem to which they are applied. For example, some comparative studies have shown that neural emulation algorithms can be inferior to the methods they were devised to replace (Taber and Deich 1988). However, it is worth remembering that these and nearly all other software programs are currently designed to run on von Neumann computers. The real power of connectionist models will ultimately result from providing an escape from such serial machines.

Of the two practical approaches to intelligent processing enhancement mentioned in this section, the hybrid approach was selected to test the potential of intermingling data with rules in a compact modular fashion. It was chosen because its design can be small and straightforward, because it can integrate data and rules in a fine-grained fashion, and because it has more potential for providing these features in a fast, dedicated, neural device than more complex schemes.

To summarize this section, the concept that drives this investigation is that of integrating rules and data in a fine-grained, modular processing environment that has the potential of being realized in highly parallel "neural" hardware. An approach that will co-locate these integrating elements was chosen over one that would compartmentalize them.

The Science. The model selected for initial investigations of the concept problem is the fuzzy cognitive map (FCM) (Kosko 1988). The FCM is based on well known equations, and features input from fuzzy set theory, providing for an inherent credibility meaure on its output. Its expert system capabilities have also been demonstrated (Taber and Siegel 1987).

The FCM is a single layer net from the family of unsupervised learning - feedback recall neural models. It can encode arbitrary patterns

$$A^k = (a_i^k,\dots,a_n^k),\ k = 1,2,\dots,m,$$

using either hardwired or differential Hebbian learning (Kosko 1986). The topology is shown in Fig. 1.

Hardwired encoding requires that the connection strengths be initially determined off-line and selectively assigned. This encoding can be used to represent the symbolic concepts and the relationships among them. Signed values are provided in the range [-1...1] to each lateral (synaptic) connection, where fuzzy positive values represent causal increase, fuzzy negative values represent causal decrease, and zero represents no causal connection. In the basic model, once the causal values are assigned, they are not expected to change without further off-line intervention.

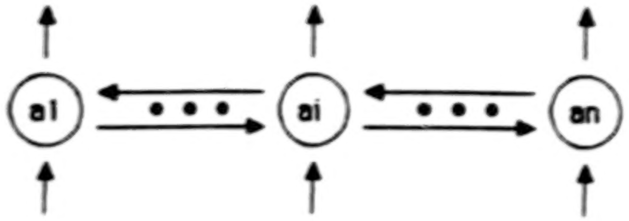


Fig. 1. The topology of the fuzzy cognitive map, (FCM). Input, ouput, and lateral connections are shown. The single layer Fa processing element array consists of neurons a_1 through a_n.

Adaptive encoding requires that the neural map automatically infer the connection values between patterns (data), and between patterns and concepts, using differential Hebbian learning. The learning algorithm correlates changes in processing element (neural) activations, such that only changes in activity in the same direction (either both increasing or both decreasing) will affect the connection (synaptic) weights. This activity is called concomitant variation. The encoding equation is

$$\dot{w}_{ij} = -dw_{ij} + \dot{S}(a_i^k)\dot{S}(a_j^k)$$

where w_{ij} is the connection strength from the *ith* to the *jth* neuron, a_i^k and a_j^k are the *ith* and *jth* components of the *kth* inference vector A^k or alternatively the activation levels of the *ith* and *jth* neurons, and dS()/dt is the time derivative of a sigmoid function. The first term in the equation is passive decay and the second term is the differential Hebbian correlation term.

For recall, the additive STM (Short Term Memory) recall equation was chosen over the more general additive recall equation for simplicity's sake. It is of the form

$$\dot{a}_i = -ba_i + \mathrm{SUM}^n{}_{j=1}\, S(a_j)w_{ji} + I_i$$

where Ii is the *ith* component of the initial inference state or the *ith* data value. The first term in the equation is passive decay, the second term is lateral feedback, and the last term is external input.

Except for the fact that the FCM utilizes the differential rather than the signal Hebbian, its topology and function are nearly identical to the Additive Grossberg (AG) model. The AG is the simplest of a family of neural models culminating in the ART2 system. Given that ART2 is basically an *evolved* AG, this suggests a migration path for the FCM.

When encoded with concepts that are highly inter-related, the FCM does not exhibit stable point behavior, but exhibits oscillatory or limit cycle behavior instead. Limit cycles consist of two or more unique sets of neurons being repeatedly activated. The dynamics of the FCM are amenable to limit cycle stability analysis given a derivation of the Lyapunov energy function (Simpson 1988). In practice, with no further external input the activation cycle soon decays as the network becomes de-energized.

The Art. No working neural network emulation program can be produced solely and directly from the formulas given above. As a working FCM model was developed by the author based on these equations, it is helpful to see how this was done.

The first issue is the choice of sigmoid function. The following function was found to produce satisfactory results

$$S_i = 1/(1 + e^{-a_i})$$

where a_i is the activation of the *ith* neuron. When incorporated into the encoding equation, the following algorithm results

$$w_{ij} = w_{ij} + (((S_i * (1 - S_i)) * \Delta w_{ij}) * ((S_j * (1 - S_j)) * \Delta w_{ji})) - (w_{ij} * d)$$

where w_{ij} is the synaptic strength from the *ith* to the *jth* neuron, Δw_{ij} is the change in w_{ij} over the last unit time period, and *d* is a decay term.

Another issue is the training method. As, under training, encoded patterns that are not continuously reinforced tend to decay, it is preferable to present the patterns in an interleaved fashion rather than in batch mode. If patterns **a**, **b**, and **c** are in the training set, they are input for encoding as **a,b,c,a,b,c,a,b,c** rather than as **a,a,a,b,b,b,c,c,c**. Furthermore, unless the synaptic connections affecting **a**, **b**, and **c** are clamped after training, to train on a further pattern **d** would require resurrecting these earlier patterns for training as well.

Decay terms must also be set. Too great a rate of decay and the neural net never develops enough energy to activate categorizing neurons or to fire rules. Too low a rate of decay and the neural net becomes overheated, activating and firing any number of neurons that bear only a weak relationship to the knowledge being recalled. In practice it was found that decay factors in the range of 0.1 to 0.2 were most suitable.

As training is mediated by a sigmoid function, synaptic weights eventually approach asymptotic values. It is often the practice to stop training when the rate of change of a weight falls below an arbitrary value, ϵ. The results reported later in this paper were achieved with an ϵ of 0.001.

To simulate simultaneous updating of synaptic and activation values (simulated parallel behavior), the new values and new delta values are found for the entire neural network before the updating of any neurons or synaptic connections occurs. This keeps the neurons from affecting one another until the entire network is ready to move. In contrast to "instantaneous" updating, spreading activation would

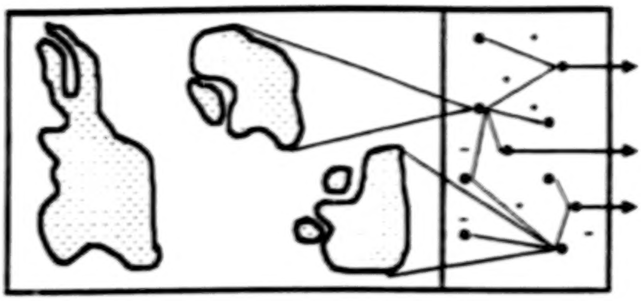


Fig. 2. The outline of a model neural net in computer memory. Dispersed data patterns (left) are mapped to concept neurons (right). Fuzzy rules are represented by the inter-relation of concepts.

produce different and less predictable activity in the network. Although spreading activation is a valid approach and is sometimes used, it was not implemented in this study.

Finally, using a discrete, serial algorithm to simulate the passage of time is no more than declaring each iteration of the network (a single training or activation pass) to be a time "unit". The change in synaptic weight or in neural activation is represented by a delta value, as found by the equations in the previous section. The time derivative value is this delta value, with respect to a single iterative time unit.

Operation. The operation of the above system is fairly straightforward. A network of processing elements or neurons is mapped out and cast in computer memory. A certain predetermined portion is then set aside for the data driven elements. Data elements can be visualized as occupying this space from top to bottom. The remainded of the network is reserved for concepts and rules (Fig. 2).

All neurons are initialized to a base or resting activation state of zero. A vigilance parameter is set to detect neurons with significant activity, the level of significance being given by the vigilance value.

Concepts are elicited, and the relationships between them set. For example, if the concept audible snarl (a_i) is thought to be twice as important as the concept fangs (a_j) in implying the concept wave big stick (a_h), then w_{ih} becomes $+0.66$ and w_{jh} is $+0.33$. The available range of values [-1...1] allows for fuzzy adjustments to this rule as long as the ratio of w_{ih} to w_{jh} remains approximately 2:1 or whatever it is judged to be.

Once set, these weights are clamped while the net as a whole is trained on data sets. Training is achieved by presenting an "analog" input pattern to tne net while simultaneously turning one or more of the concepts cells on. The "analog" input consists of a dispersion of data points that take on real values in [-1...1] such that a pattern is created. This pattern is then mapped (input) to the corresponding neurons in the net. Activated pattern neurons reinforce their relationship to various degrees with the firing concept neruons, until the rate of change falls below ϵ. This operation completes the linking of concept neurons to data neurons.

After training, many of the concept neurons may be thickly connected to dispersed areas of the data portion of the neural map. However, some of the concept neurons may only be connected to other such neurons and not at all to data cells. These neurons can be activated directly by conceptual input if they are input cells, but only indirectly by data acting through other concept neurons if they are not.

During recall, input occurs to various neural elements in the network. Typically, a continuous and changing (time variant) "analog" data pattern is read into the net, while certain concepts may be simultaneously turned on or off. Settling of the network occurs continuously as input is read in. This activity is represented in Fig. 3, where both types of input are shown.

Reporting neurons can be of various types, depending on how the neural net was trained. Some models, such as the AG,

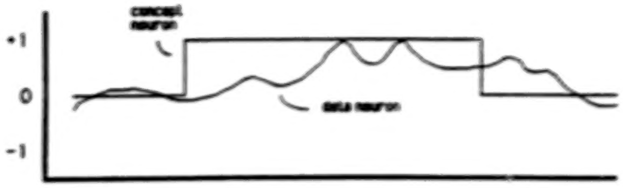


Fig. 3. Recall input to a model neural net. Concept neurons are either switched on or off, the activity of a data neuron fluctuates as real input data arrives at the system.

utilize a winner-take-all approach (a max function) to select the one cell whose output will be recognized. The system described here currently uses a vigilance threshold to detect all firing neurons over that threshold.

Test programs have been run on a Sun-3 workstation with enough memory to load a medium-sized universally interconnected net, but with no floating-point hardware. Speeds of approximately 12K IPS (interconnects per second) were achieved on an unloaded station.

Results. The system just described has been exercised via a test program based on a hypothetical space vehicle problem. The devised task is to orient an object in space such that optimal mission-sensitive behaviors can be maintained. To simplify things, the problem space was reduced to the object's purported vulnerability to particle bombardment, without regard to type or source. The problem space consists of a few identifiable surface structures on the object, a few internal operational characteristics, and a surface mapping of the object's "skin", with sufficient sensors to detect arriving particles such as micrometeorites. It should be possible under this scenario to train a neural module of the type discussed here to "recognize" vulnerable portions of its host's surface, to rank these areas in order of importance, and to offer suggestions as to appropriate orientational responses for any given micro-impact situation. If we recall the underlying thesis of this study, that of hardware realizability, the above response should realistically occur in nanosecond time frames.

The test set for which the results are presented below was based on a small rule set and the following concepts: power (available, unavailable), mission critical communications (occuring, not occuring), and the sensitive surface structures camera lens, receiving antenna, and solar cell array. These concepts and the relationships among them were knowledge engineered into the system (as described in the section Operation). Following this, the system was trained to recognize the surface structures by turning on concepts while the "sensor" - neural net component was "bombarded" with time and space variant real-valued impact patterns. In such an artificial situation care was taken to ensure that the bombardment was not random, but massed on certain areas of the sensor (data) component of the neural system. In this way, learning could occur.

Once the system was trained, test recall could be performed. Fig. 4 presents the results of one such test. An initial single spike was delivered to the system at time t = 0 centered on, but not coincident with,

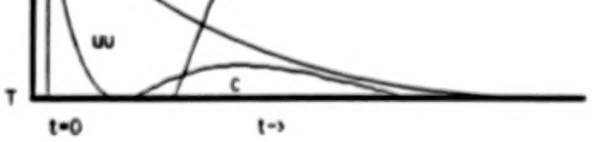


Fig. 4. The results of activating a model neural net. An activation spike **a** energizes cells that have been trained to a concept **c** (**eu**) and cells that have not (**uu**). The response cell **r** is activated as a result of rule firing within the system. T represents the activation (**A**) threshold, t is time. The figure is to scale, units are not shown.

one of the surface structures, camera. The activated neurons that represent this area of the net begin to immediately decay, the cells trained on the camera falling off more slowly than those that are not. Soon, the concept neuron representing the target concept, camera, is activated, and crests just slightly over the threshold. Almost as quickly the attached rule fires, provoking the response neuron **r** to activate. Within a 100 or so time iterations, most activity has subsided below threshold, and the system returns to its normal resting state.

In another test, input was provided over a period of time rather than as a single event. The results were similar to those above except that as more energy was being input to the system, the vigilance threshold had to be raised to mask the activity of neurons fired by weak intermediate connections (rules) in the system.

The above tests were performed with input concepts switched off. Tests run with an input neuron switched on also produced expected results– the appropriate rules fired and the correct response (asserted) neurons reported output. However, the time when the concept input neuron was switched was important. If switched on too early or too long, the neural system became energized around this neuron, such that the system became overly sensitive to rules that had this neuron as a component. If it was switched on too late or not long enough, the relevant rule would not fire. One way to adjust for this observed behavior is to reconfigure the rule(s) or the decay rate to conform to the desired level of sensitivity.

In summary, the results show that it is possible to integrate data and rules in a single neural net and achieve expected outputs. Given the small scale of these tests, however, the real problem of scaling the system to operational sizes remains unanswered. There are two issues that have a bearing on the question of scalability, and they are addressed in the next section.

Conclusions and Forecast. The evidence, albeit preliminary, seems to suggest that the implementation of the problem concept put forward in this paper is feasible. Pattern recognition and some other forms of sub-symbolic data processing are well known strengths of neural networks. And, there is enough experience with heuristic processing to feel comfortable with small sets of rules that can be easily understood by one person. Bringing them together in some fashion should draw from the strengths of both without incurring any of the major problems of either.

The issue of scalability is a factor, however. If necessary, scaling up may be done uniformly, by merely extending the neural model to ever greater numbers of neurons and IV's (interconnect values). Or, growth can be achieved through a system of interrelated, interconnected subcomponents, which need not be similar to one another.

For the moment, neural processors can be kept tractable and the major problems just alluded to kept to a minimum by targeting projects of moderate scale. If a target device is task oriented, data mapping will be of a predetermined size and kind that reduces the eventual complexity and hence the size of the neural architectural model.

Two issues that bear on ultimate size are both derived from the expert system experience. Because of the well known cost of knowledge acquisition and management (getting it, maintaining it, and ensuring that it continues to work right), rules are best kept to a small stable set, particularly in environments that are not highly human being interactive. And, if fast, compact, easily understood, task-oriented modular systems are desireable, it does not make sense to engineer a huge number of rules. Large knowledge based systems are not fast, and they are certainly not compact.

By limiting the system to a small size, however, something must be bought in exchange. What this may be is the possibility of focusing the selected model on a simple, well defined task, such that the system's overall operation can be optimized.

Eventually it may be possible to build large neural systems and to create bushy architectures of symbols and their relationships through natural, sub-symbolic learning. Until then, there is some promise that these capabilities can be provided in a small, focused manner in fast, compact, task-driven neural modules.

References.

Kosko, B. 1986. Differential Hebbian learning. AIP Conference Proceedings: Neural Networks for Computing, J.S. Denker [ed.], p.277-282, Snowbird, Utah, April, 1986.

Kosko, B. 1988. Hidden patterns in combined and adaptive knowledge networks. International Journal of Approximate Reasoning 2(1988):377-393.

Simpson, P. K. 1988. A review of artificial neural sytems II: paradigms, applications, and implementations. May be requested from the author at General Dynamics, San Diego.

Taber, W. R., and R. O. Deich. 1988. A comparison of feedforward neural networks: fuzzy operators and acoustic ship signatures. In review, International Journal of Intelligent Systems, June 1988. May be requested from the authors at General Dynamics, San Diego.

Taber, W. R., and M. A. Siegel. 1987. Estimation of expert weights using fuzzy cognitive maps. IEEE International Conference on Neural Networks II(1987):319-329.

A Connectionist Model for Dynamic Control

Kevin C. Whitfield Sharon M. Goodall James A. Reggia [†]

Department of Computer Science
University of Maryland
College Park, MD 20742

1. INTRODUCTION

This report describes the application of a connectionist modeling method known as competition-based spreading activation to a camera tracking task. The work described here is one part of an ongoing research project being conducted at the University of Maryland. The overall project is exploring the potential for automation of control and planning applications using connectionist technology. The emphasis of this effort is on applications suitable for use in the NASA space station and in related space activities. However, the results are quite general and could be applicable to control systems in general.

The technology offered by connectionist methods has several potential advantages over more conventional computational methods. For example, because connectionism is based on the tenet that useful computation may arise as an emergent property of local interactions between the nodes of a network, most connectionist models are suited for parallel processing implementations. Such parallel implementations may result in substantial reductions in processing time requirements over more conventional sequential implementations. This characteristic is desirable in real time applications, such as those intended for the space station. In addition, many connectionist models have a large degree of fault tolerance, which provides for graceful degradation in performance; only a partial loss of capability is experienced when operating in a defective condition. This property is essential to systems which operate remotely or in locations which are hard or costly to reach.

In the arena of connectionist models, the competition-based activation method provides several advantages over other techniques [5]. Of practical significance is the fact that with this technique there is often a substantial reduction in the number of links required for implementation. In many connectionist models, competitive interactions between nodes are implemented by using inhibitory (negatively weighted) connections. However, with competition-based spreading activation, competitive interactions are implemented through the competitive allocation of node output. Thus no inhibitory connections may be needed to implement competition between nodes in a network using this method.

Competitive activation mechanisms have recently been developed successfully in a number of applications. For example, they have been used to implement a system for medical diagnosis in the combined domains of neurology and psychiatry [4, 7]. This connectionist model usually identifies the (Bayesian optimal) set of disorders with the highest posterior probability for a given set of manifestations. A competitive activation method has been used to implement a 2000 node, 12000 connection system which performs the transformation of a printed word into a corresponding phonetic representation [6]. Finally, in the field of approximation theory, a connectionist model using competitive activation has been used to obtain approximate solutions to the NP-hard minimum vertex cover problem [3]. This approach yielded high accuracy and significantly outperformed a more conventional "greedy" approximation algorithm.

The goal of this work was to develop a prototype dynamic control system for camera tracking. The motivations for this task were a) to investigate the applicability of competition-based spreading activation to this general problem area, and b) to test the capabilities of the MIRRORS/II

Acknowledgements: Supported in part by NASA Award NAG1-885 and by NSF Award IRI-8451430 with matching funds from AT&T Information Systems.

† Also with Dept. of Neurology (UMAB) and University of Maryland Institute for Advanced Computer Studies.

software environment for supporting this work. This is the first application of competitive activation mechanisms to a dynamic control system.

The next section describes the problem scenario. Section 3 details the connectionist representation of the problem. It contains a description of the node sets involved, the connections between the node sets, and the activation methods used with each node set. Section 4 then describes the MIRRORS/II and CRYSTALS utilities used for implementation, and gives an example of simulation output from the connectionist model. Section 5 details the model's evaluation, along with the results. Finally, Section 6 summarizes the research results and offers suggestions for future research in this area.

2. THE CAMERA TRACKING PROBLEM

The connectionist model developed in this research is that of a simplified camera tracking controller. The system contains multiple cameras which move on a linear track and are used to photograph a field of oncoming targets. The objective of the control system is to minimize the number of targets "missed" the number of targets which propagate out of the target field without being photographed.

Figure 1 gives a pictorial representation of the camera tracking problem considered here. In this version, there are three tracking cameras which can move horizontally along the bottom edge of a 15 x 20 array of locations (cells), each of which can potentially contain a photographic target. The targets propagate as shown from the top edge of the array (row 1) towards the bottom edge (row 20), within one of the fifteen vertical columns. The targets move downward at the rate of one location per tick of the simulation clock, simulating entities whose photograph is desired as they approach and pass under a spacecraft. When the tracking cameras determine that they should move, they move horizontally in the appropriate direction at the rate of one location per tick. In this camera tracking problem, each camera photograph covers a three row by three column field of view. Photographic targets in rows 18, 19 and 20 of the target grid are potentially within a camera's field of view (depending on the camera's column position); the cameras are situated over row 19 of the target grid.

3. THE CONNECTIONIST MODEL

This section describes a connectionist model controller for the camera tracking problem described above. The node sets are described first, followed by the connections (links) and the activation methods used. Finally, the mechanisms of camera motion and picture-taking are described.

3.1. Nodes

There are three sets of nodes: the Terrain Cell node set (Tcell), the Camera Cell node set (Ccell), and the Position Cell node set (Pcell). The possible target locations of Figure 1 are represented by Tcells as shown in Figure 2. There are 300 Tcells arranged in a 15 x 20 array. The presence of a target is represented via a non-zero Tcell activation value. Target propagation is accomplished through Tcell-to-Tcell communication of activation. An activation level of 1.0 for a Tcell indicates that a photographic target is currently located at the position represented by that Tcell.

There are three tracking cameras represented by the Ccell. Associated with each camera is a 3 x 3 field of view (FOV), which is located within the bottom three rows of the target field (rows 18, 19 and 20). Each camera moves independently along the bottom edge of the field although each camera has a weighted preference for certain Tcell columns. For Ccells, a non-zero activation value is used to indicate that a photograph was taken during the previous tick of the simulation clock.

The only node set in Figure 2 not depicted in Figure 1 is the Pcell node set (three rows in lower half of Figure 2). The Pcell nodes are used to compute the level of target activity in the area which would eventually be covered by a camera at the corresponding position. The level of activation of a Pcell indicates the level of demand for camera coverage from that location. There are three rows of Pcells, one for each tracking camera. For fifteen columns of Tcells, a camera's FOV may centered at any location along the camera edge from within the thirteen central positions.

BEST COPY AVAILABLE

Hence, there are thirteen Pcells in each row (indexed by 2 through 14).

3.2. Connections

There are links from each Tcell (except for the last row of Tcells) to the Tcell directly beneath it. Designating a Tcell's position as (row,column), a Tcell at (i,j) connects to its neighbor at (i+1,j). These links are used to propagate activations representing targets toward the camera edge. Due to the nature of the activation method for Tcells (see below), the weight on these links is largely irrelevant, although it must be non-zero. For convenience, these links were implemented with unit weights.

In order to transfer photographic target information to the cameras, there are links from Tcell nodes to Pcell nodes. There is a link with the weight CTRWT to each of the three Pcells in the column directly beneath a Tcell, as well as links with the weight SIDWT to the six immediately adjacent Pcells. Of course, these links are only present when those Pcells actually exist. For example, Tcells in the first column only have three links to Pcells: a link of weight SIDWT to the three Pcells located in column 2. Some of these links are shown in Figure 2.

In order to provide photographic target location information to the tracking cameras, the bottom three rows of Tcells have links to the Ccells. The Ccells use this information from rows 18, 19 and 20 to determine if any of the Tcells in these rows both: a) have non-zero activations, and b) are located within the Ccell's current field of view[1]. Some of these links are shown in Figure 2.

The Pcell to Ccell connections are used to communicate the coverage demands from the Pcells to the Ccells. Within each column of the Pcell set, the weights to the Ccell set sum to 1.0, the maximum possible weight on a link.

In Figure 3, the three tracking cameras are shown fully covering first the leftmost and then the rightmost portion of the target field. When positioned fully to the left, Camera 1 is centered on column 2, Camera 2 is centered on column 5, and Camera 3 is centered on column 8. To avoid field of view overlap, Camera 1 should be the only camera ever centered on any of the columns 2-4. As a result, Pcells in columns 2 through 4 of the first Pcell array row have non-zero links only to the first Ccell, with the maximum possible link weight, 1.0. Similarly, Camera 1 never needs to be centered on any of the columns 9-14, so links from the Pcells in columns 9 through 14 of the first Pcell row to Ccell 1 have zero weight. In between those positions, a linear function of column position for weights on links from Pcells in the first row to the first Ccell was used. The weighting scheme for weights on links from Pcells in the third row of the Pcell array to the third Ccell mirrors those in the first row of Pcells to the first camera. Any difference between 1.0 and the sum within a Pcell column of links to first and third Ccells was used as the value for the weight of the remaining link from the Pcell in the second row to the second Ccell. A summary of these weights is found in Table 1.

Finally, there are connections from each Ccell to each of the Tcells in the bottom three rows of the target field. This connection is used by each Ccell to inform the Tcell that its target has been photographed and that the associated activation should not be propagated to the next Tcell. This process is described in Section 3.4.

3.3. Network Activation

This section details the activation methods used by the different node sets. An activation method (or rule) is a local computation carried out by a node based on the input signals it is receiving.

3.3.1. Tcell Activation

The nodes of the Tcell set use a different activation function to output activity to each set to which it is connected. Each Tcell outputs its activation to the neighboring Tcell in the next row of the same column, across the unidirectional link to that Tcell (Tcell nodes in row 20 of the Tcell array do not connect to any other Tcells).

---

[1] In the implementation, this information was not output by the Tcells, but rather was "collected" by the Ccells using links going the opposite direction. This was done to avoid excessive use of memory to maintain copies of Tcell activations at each Ccell.

BEST COPY AVAILABLE

Tcells in the bottom three rows (rows 18, 19 and 20) also output their activation to the three nodes of the Ccell set. For both of these outputs, output on a link from a Tcell is determined solely by the Tcell's activation level; the weight on these links is always 1.0 and hence is not a factor.

Tcell activity is competitively distributed to the nodes of the Pcell set. Unlike other output from Tcell nodes, Tcell output to a Pcell is proportional to the Tcell's activation level, the weight on the link between the Tcell node and the Pcell node *and* the activation of the Pcell node receiving the Tcell output. This activation method is *competition-based* since the Pcell nodes actively compete for a Tcell's activity [5]. A Pcell node's competitive strength is determined by its activation level-link weight product, relative to its competitors. The exact output at time t from Tcell node i to Pcell node j is formulated as follows:

$$out_{ji}(t) = \frac{a_j(t)\ wt_{ji}}{\sum_k a_k(t)\ wt_{ki}} \frac{a_i(t)}{1.596d^2}$$

where $a_j(t)$ is the activation level of Pcell node j at time t and wt_{ji} is the weight on the link from Tcell node i to Pcell node j, and d is the Tcell node's distance from the photographic edge of the target array. In the special case when all of the destination Pcells have an activation level of zero, the output at time t becomes:

$$out_{ji}(t) = \frac{a_i(t)\ wt_{ji}}{1.596d^2}$$

By definition, the distance from the cameras for Tcells in row 1 is 20 and the distance for Tcells in row 20 is 1. The constant $1.596 = \sum_1^{20} \frac{1}{d^2}$ is actually arbitrary, since camera motion is determined from relative activation levels. It was chosen for convenient verification of Tcell weights, in that if all Tcells are clamped to an activation value of 1.0 when the Pcells are at rest (zero activation), the net input to each Pcell is 1.0 from neighboring Tcells.

Tcell nodes receive input from three possible sources: a neighboring Tcell, a Ccell or external input. Tcell nodes in the first row of the Tcell array receive input solely from external input. A non-zero external input indicates that a target is to be propagated down this column in the Tcell array. All other Tcell nodes receive input from a preceding Tcell. Tcell nodes in rows 18, 19 and 20 also receive input from the Ccell set; a negative activation is sent to a Tcell node when a Ccell node takes a picture of that Tcell[2]. A Tcell node i updates its activation at time $(t+1)$ using the following function:

$$a_i(t+1) = \begin{cases} in_{Ccell} & \text{if } in_{Ccell} \neq 0 \\ 1.0 & \text{if } in_{Tcell} > 0 \\ 0.0 & \textit{otherwise} \end{cases}$$

where $in_{Ccell} = \sum_{i \in Ccell} in_i$, $in_{Tcell} = in_{ext} + \sum_{i \in Tcell} in_i$.

The Tcell activation function can be thought of as a simple threshold function if the negating Ccell input is ignored. If the Tcell input is above the threshold level zero the Tcell assumes its maximum level of activation of 1.0 during the next tick. If the input value is at or below the threshold, the Tcell assumes its minimum level of activation (0.0) during the next tick of the simulation clock.[3]

3.3.2. Ccell Activation

The activation of the Ccells is used simply to indicate that a picture has been taken. Each camera cell receives input from the bottom three rows of Tcells, and from a row of Pcells. If any of the Tcells in the camera's current field of view are active, the camera takes a picture of these targets. This is indicated by a positive Ccell activation level.

$$a_j(t) = \begin{cases} 1 & \textit{if photograph taken} \\ 0 & \textit{otherwise} \end{cases}$$

When a camera cell is active, it outputs negative activity to each Tcell in its current field of view using the following function:

$$out_{ij}(t) = \frac{-a_j(t)\ d^2}{(d+1)^2}$$

---

[2] The negative activation is not actually *sent* by the nodes of the Ccell set to the Tcell nodes; see Section 3.4.

BEST COPY AVAILABLE

where $a_j(t)$ is the destination Tcell's current activation and d is its distance from the photographic edge of the target array. Each Ccell also receives input from a row of Pcell nodes in the Pcell array. This input is used to determine whether to change the camera's position and if so, in which direction. This processing is described in Section 3.4.

3.3.3. Pcell Activation

Pcells have relatively simple output and activation update functions. Each Pcell outputs its current activation to a single Ccell on each advancement of the simulation clock[3]. Its activation level is determined by its input from the nodes of the Tcell set:

$$a_j(t+1) = \sum_{i \in Tcell} in_i(t)$$

3.4. Ccell Processing

In order to determine if the camera's current position should be modified, a Ccell identifies the Pcell among those to which it is connected with the highest Pcell activation-link weight product. The camera then moves one location horizontally towards that Pcell's position, if it is not already there. Since this process occurs during each tick, it follows that the maximum camera speed is one column position per tick. In the implementation, each Ccell actually looks at each neighboring Pcell's activation level and calculates this product, since the Ccell also needs to know the position attribute of the Pcell with the highest activation-weight product to determine which direction to move.

During the output portion of a simulation tick, each Ccell searches its current 3 x 3 field of view to check for target activations. If at least one target is discovered, the camera will take a photograph during that tick. In addition, the Ccell modifies the current activation value of any positively activated Tcells in the field of view to be $-\frac{1}{1.596\,(d+1)^2}$, where d is the distance associated with the activated Tcell. Using this modified activation, the Tcell will subsequently output contributions approximately equal in magnitude to those contributions made for that target during the previous tick, but with a negative sign. This output is intended to have the effect of "cancelling" Pcell activation attributable to that target during the previous tick. In addition, the resultant negative sign of the Tcell activation inhibits propagation of the target to its Tcell neighbor, causing the target to disappear after it has been photographed.

Note that this process must be done *before* Tcell output for that tick is performed. If it were done *after* Tcell output, there would be no way of halting the target propagation, since it would already have been propagated. Thus, it is required that, for the output phase of a simulation tick, processing will sequence through the node sets one at a time. Within the individual sets, of course, nodes are processed in parallel. This does not violate the parallel nature of the network, in that the processing time requirement for this processing still remains independent of the network size (assuming that the node set sizes are changed proportionately). This aspect of the camera processing suggests a performance improvement which will be addressed later in this paper.

4. IMPLEMENTATION AND TESTING SCENARIO

This section briefly describes the implementation of the connectionist network described in the previous section. It then describes network testing and performance measurements.

4.1. Implementation

The camera tracking connectionist network was implemented using the general-purpose network simulation system MIRRORS/II [1]. In addition, a high-level front-end processor for MIRRORS called CRYSTALS was used [2].

The basic input to the MIRRORS simulator consists of a network specification and a control specification. The network specification details the node sets and their members, along with various default node parameters for the set (maximum activation level, decay rate, etc.). This is followed by a description of node-specific information (including connections emanating from a node, overrides to the default parameters, etc.). The second section of input to the MIRRORS simulator is the control specification. This details such things as the length of the simulation, a schedule of external influences (called *events* in MIRRORS) that take place during the simulation, and the items to be recorded during the simulation. It also specifies in what order the node sets are to be updated and output.

---

[3] This activation value is not actually *sent* to the Ccells in the

Even for large networks, the control specification is usually generated relatively easily by hand. However, manual generation of a network specification quickly becomes tedious for reasonably large networks. This motivated the development of the CRYSTALS preprocessor, which allows the user to use predefined high-level, topologically regular structures, such as lines, spheres, etc., to specify network components. CRYSTALS also gives the user access to Franz Lisp functions to perform such things as iteration during the process of network generation. An example showing portions of the CRYSTALS-generated MIRRORS input file used for this research appears in Appendix I of this paper. The contents of this appendix corresponds to the network simulation described in Section 5.3 of this paper.

4.2. Testing Scenario

Each of the test runs made for this research had a duration of 1000 ticks of the simulation clock. During a simulation, photographic targets were randomly generated at various constant rates (densities) in the first row and propagated toward the cameras. Targets were randomly generated according to a uniform distribution; an identical random seed was used for each test run. When a target was photographed, a message recording the event was output to the screen. In a similar manner, a message was output indicating that a target was missed if that target left the last row without being photographed. The counts of the two types of messages comprise the performance measurements.

5. RESULTS

This section details the individual tests run with the MIRRORS implementation of the connectionist model. Included are the test specifics, the data collected, and a brief discussion of what the results of each test mean.

5.1. Link Weight Determination

The first set of test runs were done to collect data for comparison between different values of the parameters CTRWT and SIDWT, which are the weights on the links from the Tcell node set to the Pcell node set (see Section 3 above). This was done to empirically determine good values for these parameters, which would be utilized in further testing. All of these tests were done with an incident target density of 8%. The results from these tests are shown in Table 2.

As can be seen from this table, performance was best with the links from the Tcells to the Pcells all equally weighted (SIDWT = CTRWT = 0.333). These were the values for CTRWT and SIDWT used in further testing.

5.2. Initial Network Performance

After selecting values for CTRWT and SIDWT parameters, a set of tests was run to measure network performance under various target densities (from 1% to 50%). The results of these tests are shown in Table 3. The performance results vary from a low of about 83% with a 50% target density, to over 99% with a 1% target density. Considering the fact that the success rate for statically positioned cameras would be around 60% at any density, these performance results appear to be a substantial improvement over statically positioned cameras. There would, however, be some improvement with randomly moving cameras over statically positioned ones, due to the three row thickness of the camera field of view.

---

implementation; see discussion in the following section.

BEST COPY AVAILABLE

5.3. A Performance Improvement

Even though the performance results given in Table 3 are good, there is a way to obtain even better performance. It lies in the way in which the node sets are ordered in their respective update and output cycles. In the simulations of the previous test, the Ccells performed their processing using the Pcell activations from the *previous* tick. If, instead, this processing could access more recent Pcell activations, one would expect a performance improvement, since the propagation delay would be reduced by one tick.

This technique was used in the second set of test runs. Using the MIRRORS Order command, the specification file was modified to ensure that, in any tick, the Ccell node set update processing occurred after both the Pcell and the Tcell update processing (see Appendix I). Camera motion processing was then modified to refer to the most recent Pcell activation values.

This sequencing does not destroy the applicability of a parallel implementation for this model, as the processing time requirement still remains independent of the network size (assuming that the node sets are increased in size proportionately). The only effect is that each tick of the simulation clock must effectively be divided into two phases rather than a single phase. The results of this set of simulations is shown in Table 4. In the table, the improvement is shown in the change of the percent of photographed targets from the corresponding percent photographed given in Table 3. The results in Table 4 indicate that the modified network performed better in every test case. Indeed, at the lowest density measured, a perfect record was achieved. Because the modified network was an obvious improvement, it will be used for comparison in the remaining tests, which involve degraded performance.

5.4. Fault Tolerance

Once a reasonable network performance had been obtained, the model's performance under partially disabled conditions was studied. Since the Ccells were deliberately designed to be localized in their motion, it was expected that incapacitating one of them would drastically reduce performance. This was not explored during this research. Instead, an investigation into the fault tolerance of the network in the face of Pcell incapacitation was undertaken. Several test simulations were run with various subsets of the Pcell node set disabled (clamped to zero). These tests are outlined in the following subsections.

5.4.1. Test One

In this set of runs, all of the "even-columned" Pcells—those Pcells in columns 2, 4, 6, 8, 10, 12, 14—were disabled. This represents a 46% operational Pcell node set—a substantial level of degradation.

From Table 5 it can be seen that there was, as a result of the highly incapacitated Pcell node set, a substantial degradation in the performance. However, it must be noted that in disabling the Pcells in columns 2 and 14, all information concerning targets in columns 1 and 15 was lost. This corresponds to around $\frac{2}{15} = 13.33\,\%$ of all incident targets. Thus, even if *all* targets in columns 2 through 14 were photographed, one would not expect performance much above 86.7%. Some number of targets in columns 1 and 15 must have been photographed serendipitously.

Of course, one way to ameliorate this high degree of degradation would be to have 15 rather than 13 Pcells in each row. However, since having a camera centered on column 1 or 15 implies only having a 2 x 3 field of view on the target field, additional processing may be needed to avoid those positions when operating under normal (nondegraded) conditions.

Another interesting aspect of this set of tests is that performance degraded more at lower densities. One possible explanation depends on a property of the target generation formula: runs using a higher target density include all those targets found in runs using lower target densities, plus some additional targets. It could be that some targets which turn out to be difficult to photograph at lower densities are easier to capture with the additional targets at the higher densities.

BEST COPY AVAILABLE

5.4.2. Test Two

In this set of test runs all of the "odd-columned" Pcells—those Pcells in columns 3, 5, 7, 9, 11, 13—were disabled. This was done for two reasons. Firstly, it avoids the difficulty discovered in the previous test scenario in that the loss of those columns does not obliterate the presence of certain targets from the Pcells. Also—considering now the proposed solution to the problem identified in the previous test set—this scenario corresponds to the case where not only odd-columned Pcells are disabled, but all of the proposed additional Pcells are incapacitated as well. Thus this set of runs is applicable both to the network as it currently is, and as it would be if it were modified. For the current network, this represents a 54% operational Pcell node set. The results of these runs are included in Table 6.

As can be seen from the table, the high level (46%) of incapacitation of the Pcells has resulted in little or no degradation in the system's performance. Not only is the performance loss due to "invisible" targets no longer present, but in one case performance was actually improved. One possible explanation for a performance increase is that for some targets, while incapacitation of the even-columned Pcells made that target harder to photograph (in that it registered one column farther from the camera than it would normally), incapacitation of the odd-columned Pcells made that target easier to photograph (in that it registered one column closer to the camera). In any case, it seems that this improvement is probably coincidental; additional runs with different random number generator seeds are needed before any conclusion can be reached.

5.4.3. Test Three

In this final set of test runs, the Pcells were incapacitated in a "good-bad-bad" sequence (i.e., the Pcells in columns 3, 4, 6, 7, 9, 10, 12 and 13 were disabled). This avoided "invisible" targets while bringing the Pcell node set down to a 38.5% operational state. The results of these runs are shown in Table 7.

As can be seen, the level of degradation in system performance is only slightly larger than that which was present in the previous set of runs (see Table 6). Considering that over 61% of the Pcells are inoperable, this seems to represent a significant degree of fault tolerance.

6. CONCLUSIONS AND FUTURE RESEARCH

When fully operational, the connectionist model performed well with respect to the problem addressed. The fact that it performed so well suggests that competition-based connectionist models may have potential for control applications. This is supported by the fact that the model performed well under a range of degraded conditions. In the area where performance was lowest, feasible alternatives were found which would enhance performance. Of course, only specific types of degradation were tested — it would be advantageous in designing an actual hardware implementation to orient the design so that the probability of other more catastrophic failures is minimized. This property of selective, tolerant degradation is essential in the design of systems which are targeted for remote operation.

The application addressed in this research was admittedly of limited scope, both in the size of the problem, and in the detail of the implementation. However, the intent was not to design a fully operational system — it was simply to investigate the applicability of competition-based connectionist models to control applications. In this respect the goal was achieved, in that no problems developed which could be attributed to the connectionist approach or to the competition-based spreading activation method.

To fully investigate the degree to which this solution is appropriate to the problem addressed the design and simulation of a more conventional algorithmic solution would be needed. This would be useful at least for the purpose of comparison. In addition, an expansion of the size of the target field, along with a reduction in target density would make the problem more closely approximate real world problems — e.g. satellite reconnaissance. From the theoretical aspect, much work remains to be done. Perhaps the problem could be viewed as a nonlinear optimization problem, where an attempt could be made at deriving local minimization methods through which an appropriate global minimization goal could be approximately achieved. Also, an attempt to derive bounds on optimum performance could be made.

Currently, we are analyzing a more challenging formulation of this problem. Instead of simulating cameras with a three by three field of view, we are now simulating cameras which have a one by one field of view, on the same fifteen column wide target area. In preliminary simulations, it appears that the competitive activation method performs well. Fault tolerance tests have not yet been performed.

It is hoped that this approach will be generalizable to other areas in real-time control. The NASA space station will consist of a large number of interdependent control systems — many having common properties and goals. A common approach to their design and implementation — perhaps one embodying principles of connectionism — could greatly facilitate the goal of achieving a permanent station in space.

REFERENCES

(1) D'Autrechy C. L., J. Reggia, G. Sutton and S. Goodall, "A General-Purpose Simulation Environment for Developing Connectionist Models", *Simulation* **51**(1):5–19 [1988].

(2) Goodall S., and J. Reggia, "The CRYSTALS Preprocessor for MIRRORS", Dept. of Comp. Science, Univ. of Md. College Park, MD (internal working paper) [1988].

(3) Peng, Y., "A Connectionist Solution for Vertex Cover Problems", Institute for Software, Academia Sinica, Beijing, The People's Republic of China [1988].

(4) Peng, Y., and J. Reggia, "A Connectionist Model for Problem Solving", *IEEE Trans. Systems, Man and Cybernetics* (in press) [1989].

(5) Reggia, J. "Properties of a Competition-Based Activation Mechanism in Neuromimetic Network Models", *Proc. Intl. Conf. on Neural Networks*, II:131–138 [1987].

(6) Reggia, J., P. Marsland, and R. Berndt, "Competitive Dynamics in a Dual-Route Connectionist Model of Print-to-Sound Transformation", *Complex Systems*, (in press) [1988].

(7) Wald, J., M. Farach, M. Tagamets and J. Reggia, "Generating Plausible Diagnostic Hypotheses with Self-Processing Causal Networks", *Jrnl. of Experimental and Theoretical AI*, (in press) [1989].

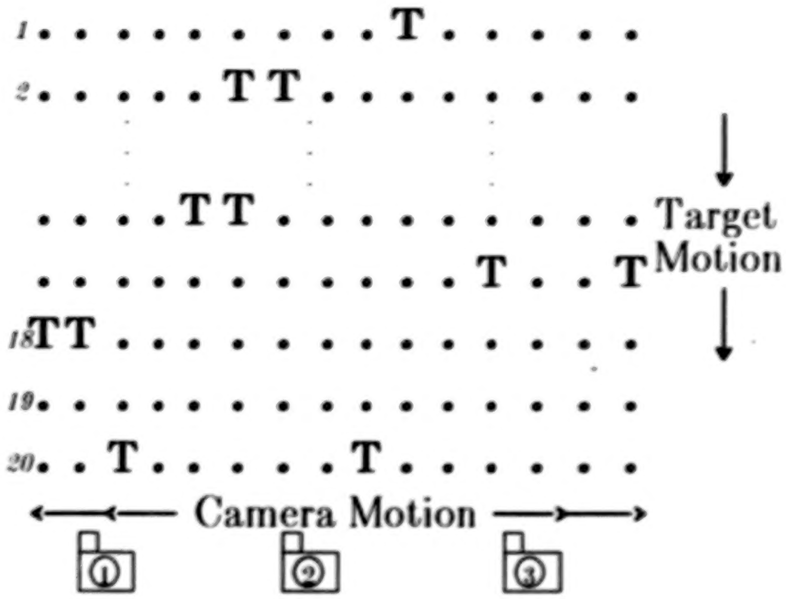

1
2
18
19
20
Target Motion
Camera Motion
1
2
3

Figure 1. Camera Controller Problem: The problem is defined by a 15 by 20 array of photographic target locations and three horizontally-moving cameras. Photographic targets progress over time from row 1 of the location grid towards row 20, staying within a column. Each camera can photograph a three row by three column area; cameras are situated over row 19 of the grid.

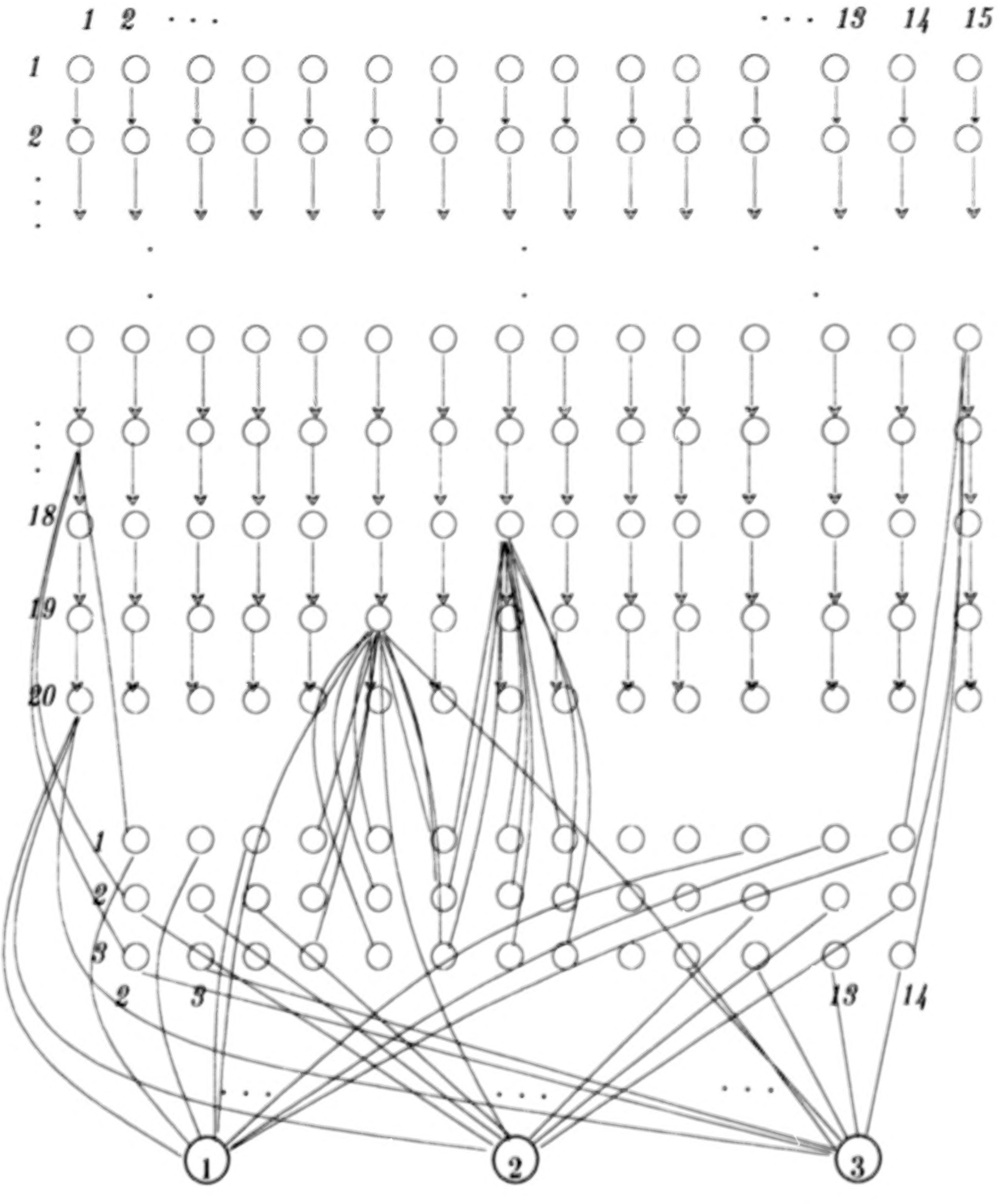


Figure 2. Connectionist Network: Rows 1 through 20 of the figure illustrate the Tcell node set and its connections. Tcells connect to their next neighbor in the Tcell set, Pcells in their column; Tcells in rows 18, 19 and 20 also connect to the Ccells. Middle 39 nodes are Pcell node set. Pcells connect Ccells. Bottom three nodes are Ccell node set. For clarity in the figure, only some of the connections between Tcells, Pcells and Ccells are shown.

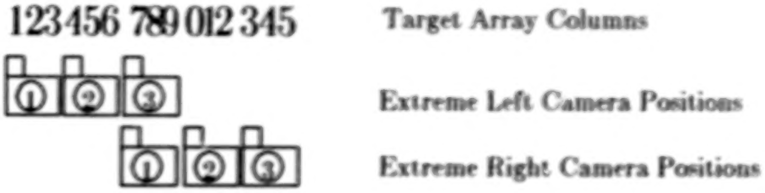

123456 789 012 345
Target Array Columns
1
2
3
Extreme Left Camera Positions
1
2
3
Extreme Right Camera Positions

Figure 3. Field of View Overlap Constraints: To avoid overlap of the three cameras' coverage, restrict their movement to be within the range of their extreme left and extreme right column positions.

Table 1. Pcell to Ccell Link Weights

Ccell Number	Column Number Within Corresponding Pcell Row												
	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1.0	1.0	1.0	0.8	0.6	0.4	0.2	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.2	0.4	0.6	0.6	0.6	0.4	0.2	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.4	0.6	0.8	1.0	1.0	1.0

Table 2. Performance Results for Various Tcell to Pcell Link Weights

SIDWT	CTRWT	Number Missed	Number Photographed	Percent Photographed
0.45	0.1	83	1079	92.86
0.35	0.3	82	1080	92.94
0.333	0.333	79	1083	93.20
0.3	0.4	93	1069	92.00
0.25	0.5	97	1064	91.65
0.2	0.6	121	1040	89.58

Table 3. Initial Performance Results

Target Density	Number Missed	Number Photographed	Percent Photographed
50 %	1218	6172	83.52
40 %	937	4950	84.08
30 %	675	3777	84.84
20 %	385	2516	86.73
15 %	221	1928	89.72
10 %	115	1309	91.92
8 %	79	1083	93.20
6 %	37	836	95.77
4 %	14	551	97.52
2 %	4	263	98.50
1 %	1	138	99.28

Table 4. Performance Results With Sequenced Update

Target Density	Number Missed	Number Photographed	Percent Photographed	Change in Percentage
50 %	873	6513	88.18	+4.66
40 %	728	5162	87.64	+3.55
30 %	493	3956	88.92	+4.08
20 %	230	2671	92.07	+5.34
15 %	131	2019	93.91	+4.19
10 %	44	1382	96.91	+4.99
8 %	36	1126	96.90	+3.70
6 %	12	861	98.63	+2.86
4 %	5	560	99.12	+1.59
2 %	2	265	99.25	+0.75
1 %	0	139	100.00	+0.72

Table 5. Fault Tolerance Results – Test One

Target Density	Number Missed	Number Photographed	Percent Photographed	Change in Percentage
10 %	185	1240	87.02	-9.90
8 %	155	1006	86.65	-10.25
6 %	125	747	85.67	-12.96
4 %	76	489	86.55	-12.57
2 %	36	231	86.52	-12.73
1 %	21	118	84.89	-15.11

Table 6. Fault Tolerance Results – Test Two

Target Density	Number Missed	Number Photographed	Percent Photographed	Change in Percentage
10 %	60	1366	95.79	-1.12
8 %	44	1118	96.21	-0.69
6 %	24	849	97.25	-1.37
4 %	6	559	98.94	-0.18
2 %	0	267	100.00	+0.75
1 %	0	139	100.00	0.0

Table 7. Fault Tolerance Results – Test Three

Target Density	Number Missed	Number Photographed	Percent Photographed	Change in Percentage
10 %	64	1362	95.51	-1.40
8 %	51	1111	95.61	-1.29
6 %	23	850	97.37	-1.26
4 %	10	555	98.23	-0.89
2 %	1	266	99.63	+0.37
1 %	0	139	100.00	0.00

APPENDIX I: Input File for MIRRORS/II

```
[set Pcell (method PCELL)(connects (Ccell oto))(size 39)
(attribute position optional dynamic (compute POSITION Ppos))]
[set Tcell (method TCELL)(size 300)
(connects (Ccell oto optional) (Tcell oto optional) (Pcell oto))
(attribute position optional dynamic (compute POSITION Tpos))]
[set Ccell (method CCELL) (connects (Pcell oto) (Tcell oto))(size 3)
(attribute position optional dynamic (compute POSITION Cpos)) ]

[implicit (member Pcell)]
[node {1,2} (Ccell ({1} 1.0)) ]
[node {1,3} (Ccell ({1} 1.0)) ]
      .

      .
[node {3,14} (Ccell ({3} 1.0))]
[implicit (member Tcell)]
[node {1,1} (Pcell  ({1,2} 0.333333)) (Tcell ({2,1} 1.0)) ]
[node {1,2} (Pcell  (/ ({1,2} 0.333333)({1,3} 0.333333))(Tcell ({2,2} 1.0))]
[node {1,3} (Pcell (/ ({1,2} 0.333333) ({1,3} 0.333333) ({1,4} 0.333333)) (Tcell ({2,3} 1.0))]
      .

      .
[node
[implicit (member Ccell)]
[node {1} (Tcell (/({20,15} 1.0)({19,15} 1.0)({18,15} 1.0)({20,14} 1.0) ({19,14} 1.0)({18,14} 1.0)
({20,13} 1.0)({19,13} 1.0)({18,13} 1.0) ({20,12} 1.0)({19,12} 1.0)({18,12} 1.0)({20,11} 1.0)({19,11} 1.0)
({18,11} 1.0)({20,10} 1.0)({19,10} 1.0)({18,10} 1.0)({20,9} 1.0) ({19,9} 1.0)({18,9} 1.0)({20,8} 1.0)
({19,8} 1.0)({18,8} 1.0)({20,7} 1.0) ({19,7} 1.0)({18,7} 1.0)({20,6} 1.0)({19,6} 1.0)({18,6} 1.0)
({20,5} 1.0) ({19,5} 1.0)({18,5} 1.0)({20,4} 1.0)({19,4} 1.0)({18,4} 1.0)({20,3} 1.0) ({19,3} 1.0)
({18,3} 1.0)({20,2} 1.0)({19,2} 1.0)({18,2} 1.0)({20,1} 1.0) ({19,1} 1.0)({18,1} 1.0)))
(Pcell (/ ({1,8} 0.2)({1,7} 0.4)({1,6} 0.6)({1,5} 0.8)({1,4} 1.0)({1,3} 1.0) ({1,2} 1.0))]
      .

      .

      .

;
; control specification
;
[control ALTCONTROL]
[events (clamp)(display) (Gen_Targets (parser GENTARG PGen_Targets)
                 (handler GENTARG Gen_Targets)]
[order (Output Ccell Tcell)      ; so that Ccells modify Tcells properly
   (Update Pceil Tcell Ccell)] ; so that Ccells get recent Pcells
[Gen_Targets ? (Tcell (& {1,1} {1,2} {1,3} {1,4} {1,5} {1,6} {1,7} {1,8} {1,9} {1,10}
                {1,11} {1,12} {1,13} {1,14} {1,15})) 10.0]
[display ? (Tcell) act nil ((string (%s "Tick: ")(%d "tick"))(string (%s "Tcells: "))(perline 15))]
[display ? (Pcell) act nil ((string (%s "Tick: ")(%d "tick"))(string (%s "Pcells: "))(perline 14))]
[display ? (Ccell) act nil ((string (%s "Tick: ")(%d "tick"))(string (%s "Ccells: "))(perline 3))]
[run 1000]
[exit]
```

Development Tools/Methodologies

BLANK PAGE

MIS35N

EXPERT SYSTEM DEVELOPMENT METHODOLOGY AND THE TRANSITION FROM PROTOTYPING TO OPERATIONS: FIESTA, A CASE STUDY

Nadine Happell and Steve Miksell
Stanford Telecommunications, Inc.*+

Candace Carlisle‡
NASA/GSFC

ABSTRACT

A major barrier in taking expert systems from prototype to operational status involves instilling end user confidence in the operational system. End users want assurances that their systems have been thoroughly tested, meet all their specifications and requirements, and are built based on designs which are reliable and maintainable. For most software systems, the waterfall life cycle model can provide those assurances. However, this model is inappropriate for expert system development, where an iterative refinement approach is commonly employed. This paper will look at different life cycle models and explore the advantages and disadvantages of each when applied to expert system development. The Fault Isolation Expert System for TDRSS Applications (FIESTA) is presented as a case study example of development of an expert system. FIESTA is planned for use in the Network Control Center (NCC) at Goddard Space Flight Center in Greenbelt, Maryland. The end user confidence necessary for operational use of this system is accentuated by the fact that it will handle real-time data in a secure environment, allowing little tolerance for errors. The paper discusses how FIESTA is dealing with transition problems as it moves from an off-line standalone prototype to an on-line real-time system.

1.0 DEVELOPMENT LIFECYCLE MODELS

When discussing software development methodologies and lifecycle models, there are two which are readily seen as the most common: the waterfall model, and iterative refinement. Of these, iterative refinement is by far the most common for expert system development. In fact, some experts claim the use of iterative refinement for expert system development "seems inescapable" [1].

The basic approach behind iterative refinement is to start off with a basic concept of what the system should be like, build a prototype, and evaluate what modifications need to be made to this prototype to bring it closer to the desired system. Modifications are made, and the system is continuously reevaluated, until a satisfactory system exists. This approach is generally used because hard, fast requirements for the system can not be specified ahead of time. Basically, the developer and user must tinker with the system, experimenting to see what will work the best. An advantage to this approach is that it provides a rapid operational capability. The very first prototype could potentially be of limited use to the end user, even though it may be lacking in many areas. And each prototype provides a realistic base on which to build the next round of improvements.

There are problems to this approach however. After numerous modifications and reworking, the code tends to lose its integrity, resulting in the expert system equivalent of spaghetti code [2]. Also, quick, temporary fixes have a tendency to become permanent before long-range architectural concerns are addressed. These problems can make the system difficult and expensive to maintain.

---

*1761 Business Center Dr., Suite 400, Reston, VA 22090

+FIESTA development has been undertaken by Stanford Telecommunications, Inc. as a subcontractor to Computer Sciences Corp. under NASA contract NAS5-31500.

‡Code 532.3, NASA/Goddard Space Flight Center, Greenbelt, MD 20771.

The classic answer to these problems has been to use the waterfall model, which advocates the stagewise development shown in Figure 1. Additional steps like cost analysis, hardware studies, and feedback loops can be added, but the basic approach remains the same. In the waterfall model, the requirements must be defined before the system is designed, the design must be complete before coding starts, and the system must be thoroughly tested before going operational. This model has become the standard for most conventional software development because it provides the end user with confidence that the system meets his requirements, has been thoroughly tested, and is based upon a design which facilitates maintenance. However, most expert system applications can not complete one of the first steps, requirements definition, until several extensive prototypes have been built. And the prototypes can't be adequately tested until they are put in an operational environment.

There are software engineers who are exploring new models in order to balance out the advantages and disadvantages of these two approaches, the most notable of which is Barry Boehm's Spiral model [3]. However, this model still has some problems, and is not yet ready for general application outside the research environment. So for the present, developers are restricted to using iterative refinement, the waterfall model, and derivatives thereof.

For applications in which the expert system can be tested in an operational environment without impacting existing operational systems, the iterative refinement approach may be sufficient, even if it isn't optimal. However, there are applications, such as the Fault Isolation Expert System for TDRSS Applications (FIESTA), which can not be tested on-line without the potential for affecting existing operations, and therefore require a more rigorous approach. This paper will look at the FIESTA project and how it is trying to balance the need for an iterative refinement development approach with the end user's need for a trustworthy system before going on-line.

2.0 FIESTA

2.1 BACKGROUND

2.1.1 FIESTA/Problem Domain

The purpose of the Fault Isolation Expert System for TDRSS Applications (FIESTA) prototype is to automate fault isolation and service monitoring for the Space Network (SN). The SN, illustrated in Figure 2, provides NASA's primary means of communication with satellites and the space shuttle. It is composed of Tracking and Data Relay Satellites (TDRS), the NASA Ground Terminal, the White Sands Ground Terminal, and the Network Control Center (NCC). Data from spacecraft are relayed through the TDRS, then sent through White Sands and the NASA Ground Terminal to each spacecraft's control center. The NCC at Goddard is responsible for scheduling use of the space network's resources, monitoring the quality of the services it provides, and diagnosing problems. High-speed messages called schedule orders (SHOs) transmit the space network schedule from the NCC to White Sands. Fault Isolation Monitoring System messages, operations messages, and operations data messages are sent between White Sands, the NASA Ground Terminal, and the NCC; console operators in the NCC can use the information in these messages to monitor service quality and diagnose problems. Console operators currently do this job by viewing cluttered, number-filled displays, detecting when a problem occurs, and determining an appropriate course of action. These tasks are known as service monitoring and fault isolation, and are extremely labor-intensive.

This application of human expertise currently provides acceptable levels of service monitoring and fault isolation. However, providing for projected growth in network loading, while retaining the current levels of performance requires automating many of these functions.

Translating expertise into conventional software requirements is extremely difficult.

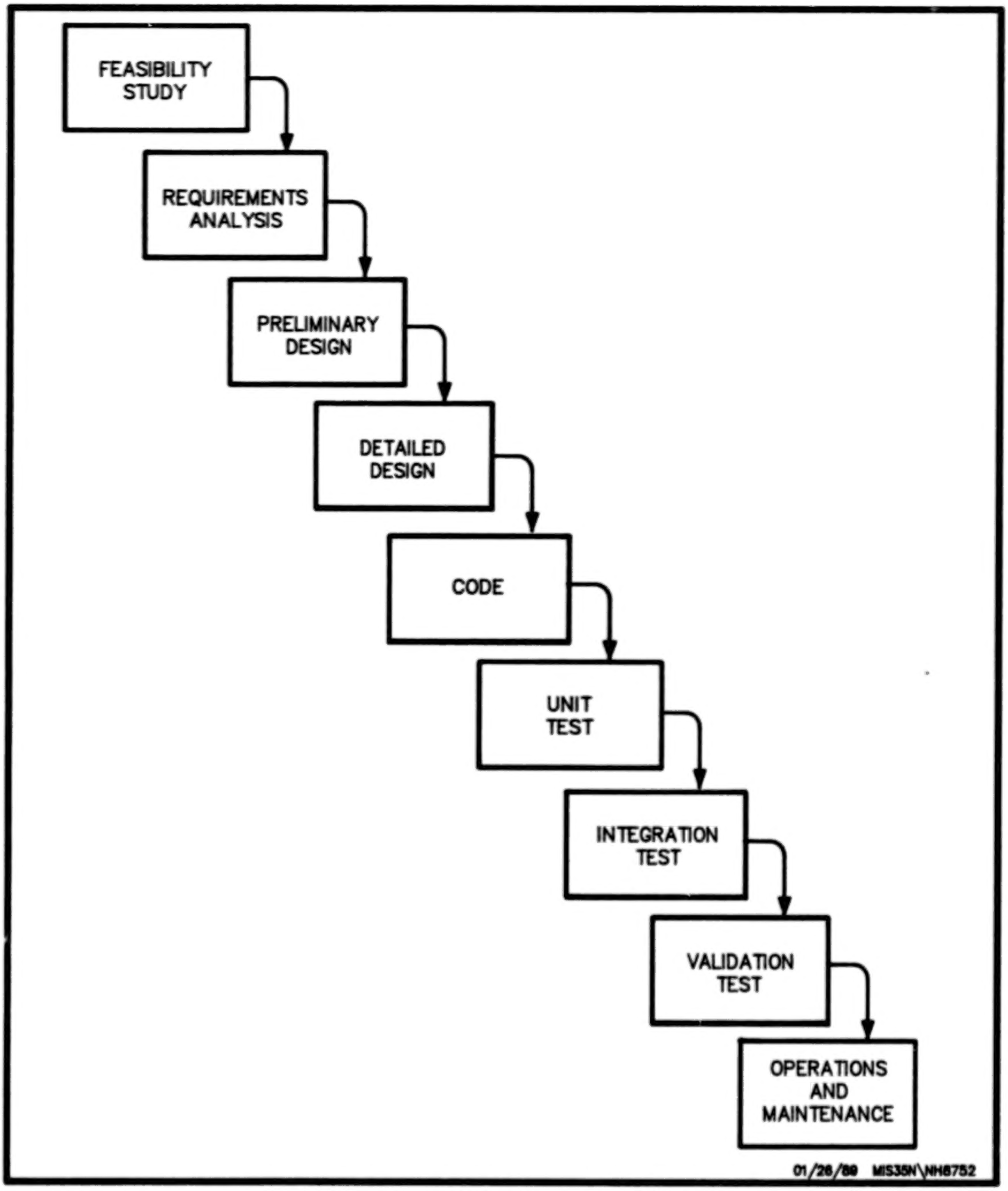

FEASIBILITY STUDY
REQUIREMENTS ANALYSIS
PRELIMINARY DESIGN
DETAILED DESIGN
CODE
UNIT TEST
INTEGRATION TEST
VALIDATION TEST
OPERATIONS AND MAINTENANCE
01/26/89 MIS35N\NH8752

FIGURE 1: WATER FALL MODEL

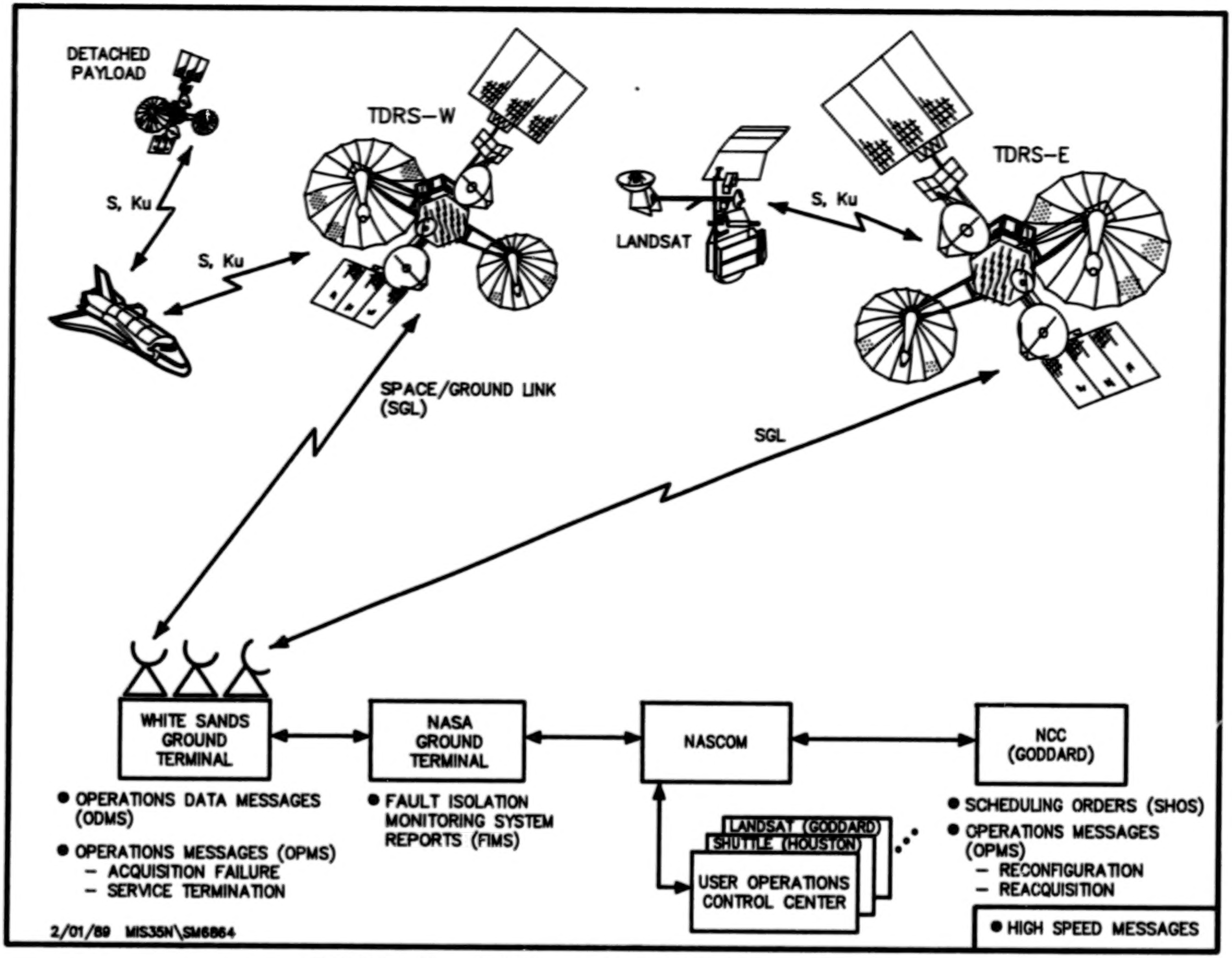

DETACHED PAYLOAD
TDRS-W
TDRS-E
LANDSAT
S, Ku
S, Ku
S, Ku
SPACE/GROUND LINK (SGL)
SGL
WHITE SANDS GROUND TERMINAL
NASA GROUND TERMINAL
NASCOM
NCC (GODDARD)
● OPERATIONS DATA MESSAGES (ODMS)
● OPERATIONS MESSAGES (OPMS)
– ACQUISITION FAILURE
– SERVICE TERMINATION
● FAULT ISOLATION MONITORING SYSTEM REPORTS (FIMS)
LANDSAT (GODDARD)
SHUTTLE (HOUSTON)
USER OPERATIONS CONTROL CENTER
● SCHEDULING ORDERS (SHOS)
● OPERATIONS MESSAGES (OPMS)
– RECONFIGURATION
– REACQUISITION
● HIGH SPEED MESSAGES
2/01/89 MIS35N\SM6864

FIGURE 2: NASA TDRSS NETWORK OVERVIEW

However, fault isolation and service monitoring are good candidates for expert system automation because a "rules of thumb" diagnostic process is used, and experts are available.

2.1.2 The FIESTA Approach to Expert Systems Automation

Since an expert system has never been used in the NCC environment, integration of a fault isolation/service monitoring expert system into the NCC was considered a high risk project. To minimize this risk, it was decided to use a 2-step prototyping approach. First, a standalone off-line prototype would be developed by Stanford Telecommunications (STel) on a Symbolics lisp machine. This prototype would receive its input from old NCC messages, archived on "log" tapes. This step would provide a proof-of-concept. Second, the expert system prototype would be put on-line in the NCC, receiving its message input in real-time. The FIESTA second stage prototype would then be used as a tool by NCC console operators.

The operators' hands-on experience would allow requirements and an operations concept for an operational service monitoring and fault isolation system to be developed.

The first stage of the prototype has already been developed and successfully demonstrated to operations personnel. The prototype has been used to diagnose old shuttle, as well as services for spacecraft such as ERBS, SME and LANDSAT. FIESTA's results have been compared to the real-time diagnoses made by the console operators. Currently, the second stage (on-line) prototype is being implemented. A System Requirements and Critical Design Reviews have been held. These reviews are typically held for all Goddard software projects.

In order to bring FIESTA on-line, the Symbolics will have to interface with the NCC's Communication and Control Segment (CCS) computer via a local area network (LAN). Existing code on the CCS computer will be modified to provide messages to FIESTA, code must be installed in the CCS to translate messages into s-expressions, and LAN interface software for the Symbolics will be written. Thus, on-line FIESTA software consists of 4 components: modified CCS software, expert system, LAN interfaces and new CCS software.

2.2 FIESTA LIFECYCLE

2.2.1 Overall Development Approach

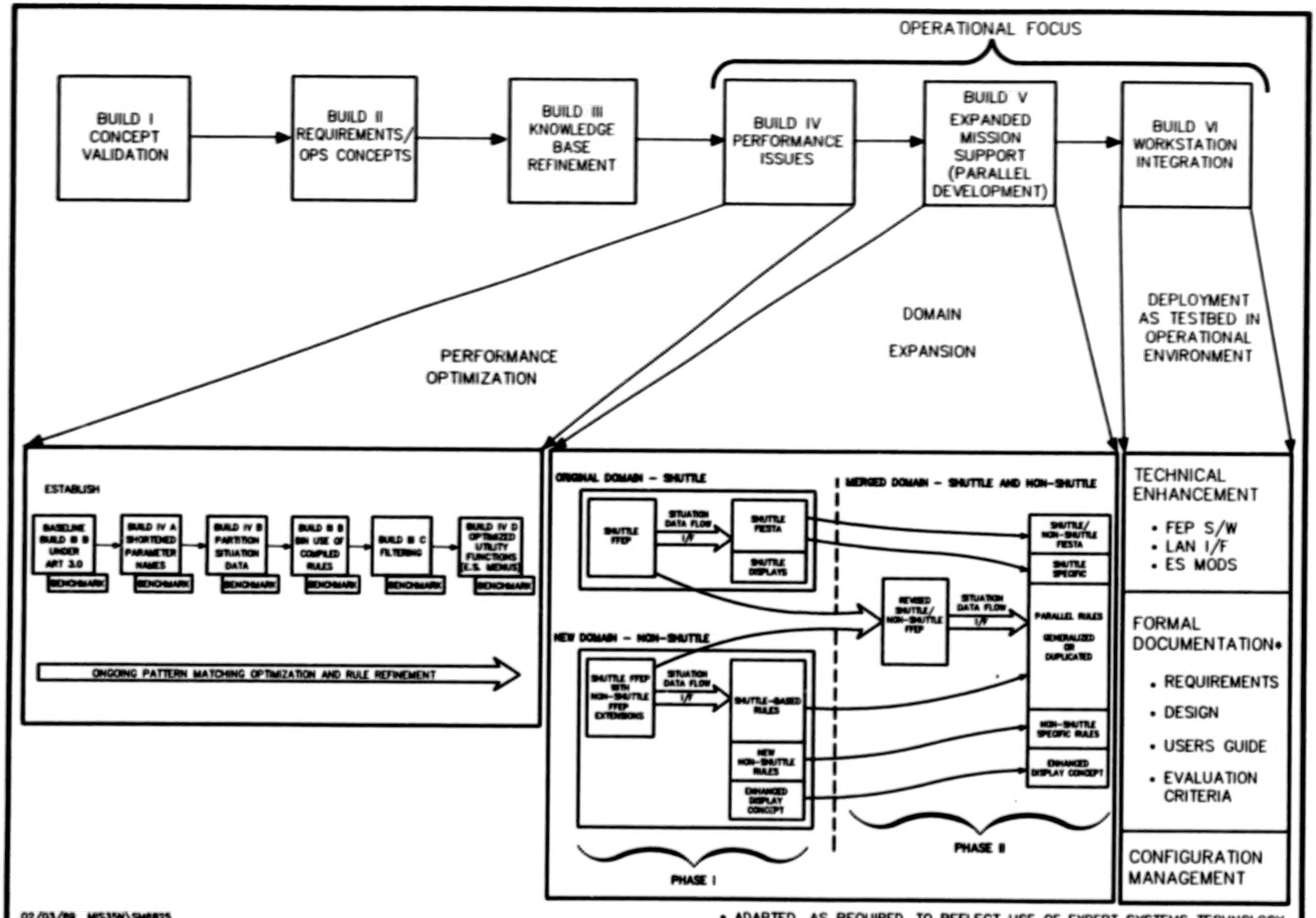
The FIESTA expert system implementation has emphasized iterative development and the refinement of requirements based on frequent demonstrations allowing operations feedback. Structure has been imposed on this process by the use of "builds". Successive builds, each with clearly defined requirements and objectives, were characterized by increasing levels of complexity. Figure 3 provides an overview of this development approach.

The first three builds served to refine the operational concept and supported the development of realistic requirements. This phase also included the establishment of the development configuration illustrated in Figure 4. Two features of this development approach have been particularly significant in the transition to operational deployment. These were (1) the use of actual (archived) data to drive development and (2) the creation of an independent front end processor (FEP) to access the archived data and emulate the NCC environment.

Use of actual data provided clearly defined requirements in terms of inputs to the system. Accessing this data via the standalone FEP (emulating the NCC) allowed expert system development to proceed as if it were actually in the target environment. For the expert system component, changes required for operational deployment were concentrated in areas where emulation differed from reality.

The final three builds focused on the transition to the operational environment. Performance Issues (Build IV) were concerned with operation in a real-time environment. Detailed results of this effort are presented in [4].

The expanded mission support (Build V) was accomplished using a replicate and merge approach [5]. This stage also provided the opportunity to restructure the rule organi


OPERATIONAL FOCUS
BUILD I CONCEPT VALIDATION
BUILD II REQUIREMENTS/ OPS CONCEPTS
BUILD III KNOWLEDGE BASE REFINEMENT
BUILD IV PERFORMANCE ISSUES
BUILD V EXPANDED MISSION SUPPORT (PARALLEL DEVELOPMENT)
BUILD VI WORKSTATION INTEGRATION
PERFORMANCE OPTIMIZATION
DOMAIN EXPANSION
DEPLOYMENT AS TESTBED IN OPERATIONAL ENVIRONMENT
ESTABLISH
BASELINE BUILD III B UNDER ART 3.0
BUILD IV A SHORTENED PARAMETER NAMES
BUILD IV B PARTITION SITUATION DATA
BUILD III B BIN USE OF COMPILED RULES
BUILD III C FILTERING
BUILD IV D OPTIMIZED UTILITY FUNCTIONS (E.S. MENUS)
BENCHMARK
BENCHMARK
BENCHMARK
BENCHMARK
BENCHMARK
BENCHMARK
ONGOING PATTERN MATCHING OPTIMIZATION AND RULE REFINEMENT
ORIGINAL DOMAIN – SHUTTLE
SHUTTLE FFEP
SITUATION DATA FLOW
I/F
SHUTTLE FIESTA
SHUTTLE DISPLAYS
NEW DOMAIN – NON-SHUTTLE
SHUTTLE FFEP WITH NON-SHUTTLE FFEP EXTENSIONS
SITUATION DATA FLOW
I/F
SHUTTLE-BASED RULES
NEW NON-SHUTTLE RULES
ENHANCED DISPLAY CONCEPT
PHASE I
MERGED DOMAIN – SHUTTLE AND NON-SHUTTLE
REVISED SHUTTLE/ NON-SHUTTLE FFEP
SITUATION DATA FLOW
I/F
SHUTTLE/ NON-SHUTTLE FIESTA
SHUTTLE SPECIFIC
PARALLEL RULES
GENERALIZED OR DUPLICATED
NON-SHUTTLE SPECIFIC RULES
ENHANCED DISPLAY CONCEPT
PHASE II
TECHNICAL ENHANCEMENT
• FEP S/W
• LAN I/F
• ES MODS
FORMAL DOCUMENTATION•
• REQUIREMENTS
• DESIGN
• USERS GUIDE
• EVALUATION CRITERIA
CONFIGURATION MANAGEMENT

02/03/89 MIS35N\SM6825

• ADAPTED, AS REQUIRED, TO REFLECT USE OF EXPERT SYSTEMS TECHNOLOGY

FIGURE 3: FIESTA DEVELOPMENT APPROACH

BEST COPY AVAILABLE

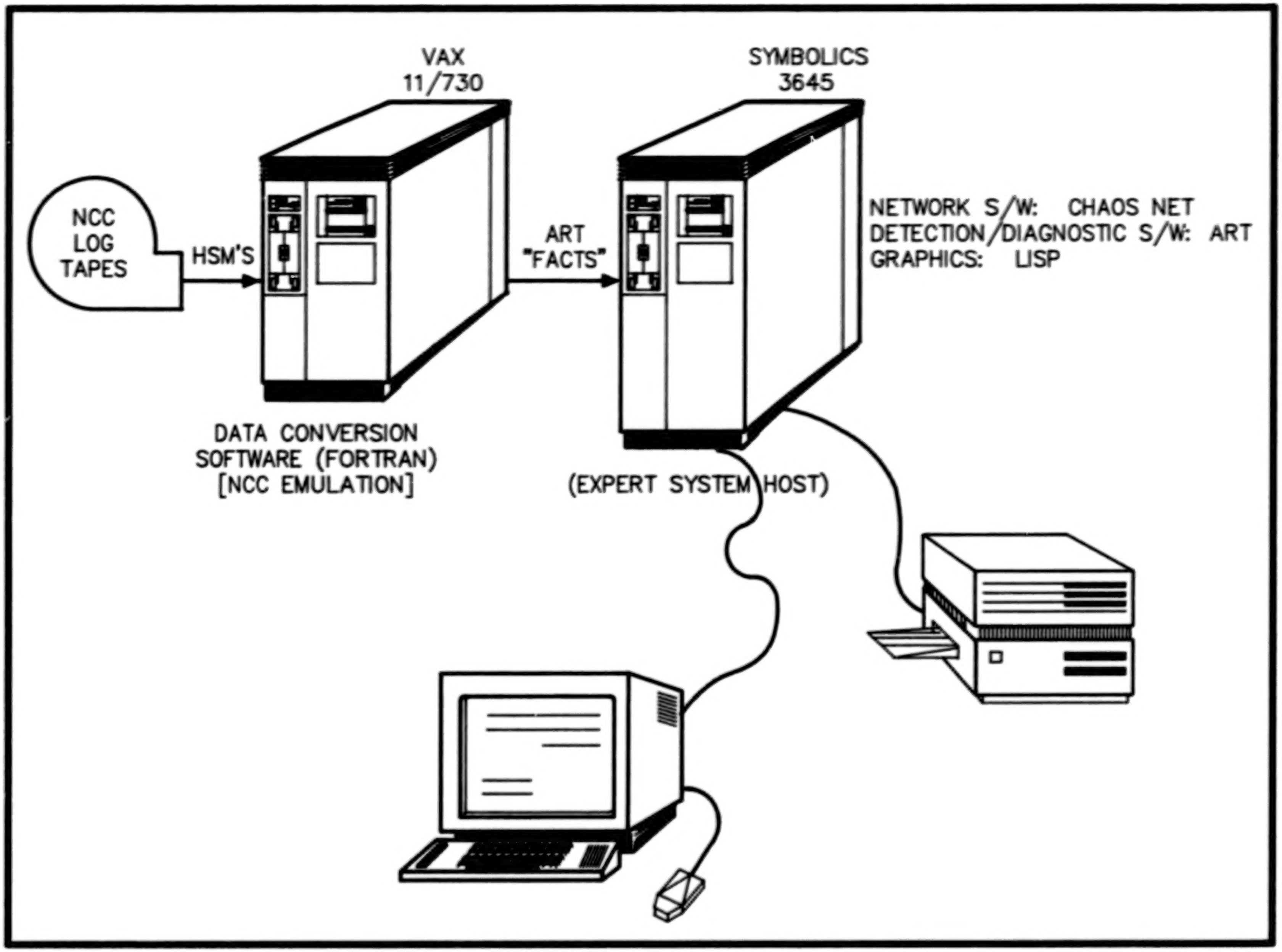

VAX
11/730
SYMBOLICS
3645
NCC
LOG
TAPES
HSM'S
ART
"FACTS"
NETWORK S/W: CHAOS NET
DETECTION/DIAGNOSTIC S/W: ART
GRAPHICS: LISP
DATA CONVERSION
SOFTWARE (FORTRAN)
[NCC EMULATION]
(EXPERT SYSTEM HOST)

FIGURE 4: DEVELOPMENT CONFIGURATION

02/03/89 MIS35N\SM8761

zation. As noted earlier, spaghetti code within an ad hoc structure often plagues development based on iterative refinement. Build V allowed some of these deficiencies to be corrected. FIESTA development is currently completing Build VI, integrating the workstation into the operational environment. This stage is discussed in the section which follows.

2.2.2 Work in Progress (On-Line Integration)

The FIESTA expert system task is currently well into Build VI, on-line integration. FIESTA is being installed as an on-line testbed to support final assessment of the concept and application of expert system technology. Despite the testbed designation, the mission-critical nature of the NCC environment remains applicable to the FIESTA implementation.

Major technical elements of the on-line integration include:

- Formal specification of functions for the On-line Testbed
- Allocation of functions to available processors
- Interface/access to the live data (LAN and CCS development)
- Modifications to the expert system component to account for "actual vs emulation" differences.

Supporting documentation that was recognized as important for the on-line operation included:

- Implementation Plan
- Transition Plan (from stand-alone prototype to on-line testbed)
- Functional Requirements Document
- Systems Requirements Document
    - Conventional format for interfaces and algorithmic processing
    - Modified format for expert system component
- Systems Design Document
    - Same distinction as for Systems Requirements
- Users Manual/Operational Scenarios
- Operations Concept for the On-line Testbed
- Evaluation Criteria.

Configuration management was also identified as a necessary element of deployment in an operational environment.

While the above items are also associated with conventional software engineering efforts, the application of expert system technology introduced the need for adaptation. In the following sections, the manner in which FIESTA methodology was accommodated are presented.

3.0 TESTBED DEVELOPMENT

3.1 FIESTA DEVELOPMENT STANDARDS

Usually, prototyping is associated with a "quick and dirty" software development approach. This approach was used for the off-line prototype. Operational software for the NCC is typically developed using formal standards, since the NCC must reliably operate in real-time, 24 hours a day. For the FIESTA on-line prototype, a balance has been struck between the formal and "quick and dirty" approaches. This "rapid software development" approach is tailored to meet the productivity goals of FIESTA, without compromising the NCC's real-time reliability. FIESTA documentation is pared down to technical details, eliminating boilerplate and redundant information. Reviews and walkthroughs are held by small working groups of technical personnel. Formal testing is limited to interfaces between the Symbolics and the CCS computer (not the expert system's knowledge base or rules). Three different levels of software development formality exist, based upon the level of risk associated with each software area.

Modifications to existing CCS software are done using the formal NCC development and Configuration Management (CM) standards. This approach is needed since CCS software is critical to the NCC's real-time operations.

New CCS software and Symbolics LAN Interface development use modified unit code, prolog, unit testing, and development certification standards. This software is of low risk since it can be deactivated, but retains some formality because it interfaces with the NCC LAN and/or resides on a critical real-time computer. Standard NCC CM techniques are employed in this area.

The FIESTA expert system has been developed by iterative refinement and demonstration. Minimal formality has been imposed on the expert system since it can simply be disconnected from the LAN should problems occur. However, system requirements and design have been documented using formats developed and tailored for the FIESTA expert system. A CM approach appropriate for the expert system component is also being developed for the Symbolics software.

3.2 DESIGN DOCUMENTATION

Even though there was no explicit design stage in the development of the FIESTA expert system component, we decided to document the design of the prototype [6]. By documenting the design we can facilitate its maintenance. We also found that in the effort to document the design, we were able to clean up the system, thus restoring most of the integrity which had been lost by numerous reworkings. The methods we used to document the design helped identify areas that needed to be cleaned up, and facilitated the restoration.

3.2.1 Structure Charts

When working with a knowledge base the size of FIESTA (600+ rules), the easiest thing to lose track of is the rule interactions. The rules most likely to interact in FIESTA are those which are related functionally. Therefore, we tried to keep functionally related rules together in the source code. However, with several developers working on the system over several years and many modifications to the system, it was inevitable that these groupings became less and less cohesive over time.

We used structure charts showing the functional hierarchy of FIESTA to help restore these groupings. In the process of shuffling rules, we were able to identify redundant rules, and rules which could be generalized to cover a higher-level function. Therefore, we were able to streamline the number of rules. In addition, we used codes to indicate which lower-level functions represented individual rules, and which higher-level functions represented different types of files of source code, and placed these indicators in the appropriate structure chart boxes. As a result, the structure charts can be used as a table of contents to the source code, identifying which functions are performed, and exactly which rules perform them. Therefore, the structure charts can be a valuable tool for future maintenance.

3.2.2 Rule-Relation Lists

We determined that a utility built into the expert system building tool (we used ART) provided an excellent means of keeping track of rule interactions. ART keeps a list of all rules, and another list of fact relations (generic fact formats). These lists are cross-referenced, so that the developer can view all of the rules which either reference a relation (using that relation as a pattern on their left-hand side), or assert facts defined by that relation into the knowledge base on the rule's right-hand side. These lists are useful in many ways, and can serve some of the same functions as data flow diagrams.

First they identify rules which could potentially impact one another. If a modification is made to a relation in one rule, the rule-relation list can be referenced to identify which other rules may be affected by that change. Secondly, they can be used to help prevent superfluous facts from being asserted into the knowledge base. If there is a relation which has no rules which reference it (use it on the rule's left-hand side), then all of the rules which assert facts defined by that relation are adding information to the knowledge base which is never used. Likewise, useless rules can be identified by examining relations which have no rules to assert facts of that type. If these facts are not asserted at system initialization, all of the rules which reference that relation will never fire, because no facts of that type are ever asserted.

3.2.3 Viewpoint Structure

The third most critical tool we used to document the expert system design was a mapping of relations to viewpoints. Viewpoints are ART-provided utilities which allow the developer to partition the knowledge base into related areas. See the design document [6] for a discussion of the viewpoint network used in FIESTA. Each relation was defined to be used at different levels of the viewpoint network. Asserting or retracting a fact from the wrong viewpoint level could have major and unexpected effects on the program's execution. This documentation most closely parallels the documentation of a database design.

Although we were unable to apply standard waterfall model methods of design documentation to FIESTA, we were able to establish means to document the design which served many of the same purposes. This documentation should aid in future maintenance efforts and help to retain the integrity of the system.

3.3 CONFIGURATION MANAGEMENT

Configuration Management (CM) has been employed in an informal manner throughout the FIESTA Life cycle. Directories are associated with each of the Builds (and sub-Builds), and changes within files have been documented with comments in the preface describing the modifications and when thye were made, as well as identifying the responsible individual(s). This approach will be further formalized for operational use to assure that the following key requirements of CM are satisfied:

- Identification and availability of all source corresponding to the operational system
- Clearly defined procedures for introducing and tracking changes in the system (i.e., a complete audit trail)
- The ability to revert to an earlier version of the system.

Furthermore, CM will operate in conjunction with Quality Assurance Procedures (modified for rapid software development) to maintain a viable, dependable product. Central to the CM approach is the logging of all update procedures and the use of existing utilities (on both the CCS development machine and the Symbolics) to identify and archive files.

SUMMARY

FIESTA development can be reasonably described as a process of iterative refinement. Use of the Waterfall Model was precluded by the very factor (i.e., ambiguous requirements associated with the automation of expertise) which supported the use of expert system technology. Nonetheless, many of the features, milestones, and particularly documents which characterize the Waterfall Model are important for the effective development, implementation and maintenance of a computer-based system. Generation of such documentation and the specification of milestones similar to those found in the Waterfall Model have characterized the transition of FIESTA from prototype to operations. Even though adaptation has been required to accommodate expert system technology, the intent of the documentation and the various reviews corresponds closely to that associated with conventional software systems.

ACKNOWLEDGMENTS

The FIESTA development described in this paper was performed by Stanford Telecommunications, Inc. as a subcontractor to Computer Sciences Corp. (CSC) under Contract NAS5-31500. NASA and CSC personnel are also providing expertise required for the CCS and LAN modifications mentioned in the paper. NASA and CSC responsibiliies and participation involves overall CCS software development; their technical involvement in the integration of FIESTA in the NCC has been essential.

The ongoing encouragement and support of Paul Ondrus, Anthony Maione and Dawn Lowe of NASA have been appreciated and are acknowledged. Roger Werking (CSC-NCC Program Manager) and his support and technical personnel also deserve our thanks.

REFERENCES

1. Hayes-Roth, F., Waterman, D.S., and Lenat, D.N., Building Expert Systems, Addison-Wesley, Reading, Mass., 1983.

2. Soloway, E., Bachant, J., and Jensen, K., "Assessing the Maintainability of XCON-in-RIME: Coping with the Problems of a VERY Large Rule-Base", Proceedings AAAI-87 Sixth National Conference on Artificial Intelligence, Seattle, Washington, 1987.

3. Boehm, B., "A Spiral Model of Software Development and Enhancement", Computer, May, 1988.

4. Wilkinson, W., Happell, N., Miksell, S., Quillin, R., Carlisle, C., "Achieving Real-Time Performance in FIESTA", Proceedings of the 1988 Conference on Space Application of Artificial Intelligence, NASA CP 3009, May 24, 1988.

5. Happell, N., Miksell, S., "Domain Expansion (with Maintenance Implications) of a Diagnostic Expert System: The FIESTA Example", In Preparation, Stanford Telecommunications, Inc., MIS845, 1989.

6. "FIESTA On-line Testbed Design - Volume II: Expert System Design", Stanford Telecommunications, Inc. Technical Report, TR880135, Oct. 1988.

FIESTA ON-LINE TESTBED DOCUMENTATION

"Functional Requirements for the Fault Isolation Expert System for TDRSS Applications (FIESTA) On-Line Testbed", TR870114, Revision B, 25 October 1988.

"Fault Isolation Expert System for TDRSS Application (FIESTA) Front End Process Requirements", TR870114.

"FIESTA On-Line Testbed Design-Volume II: Expert Sytsem Design," Stanford Telecommunications, Inc. TR880135, October 1988.

"User Manual for the Fault Isolation Expert System for TDRSS Applications (FIESTA) Testbed", MAN38ATP, 30 January 1989.

"Evaluation of the Fault Isolation Expert System for TDRSS Applications (FIESTA) On-Line Testbed", TR880137, 7 November 1988.

"Fault Isolation Expert System for TDRSS Applications (FIESTA) Implementation Plan", FIESTA Task Planning Document (C. Carlisle/532), 30 September, 1988.

BLANK PAGE

APPLICATIONS OF FUZZY SETS TO RULE-BASED EXPERT SYSTEM DEVELOPMENT

Robert N. Lea

NASA/Johnson Space Center, Houston, Texas

ABSTRACT

In this paper, problems of implementing rule-based expert systems using fuzzy sets are considered. A fuzzy logic software development shell is used that allows inclusion of both crisp and fuzzy rules in decision making and process control problems.

Results are given that compare this type of expert system to a human expert in some specific applications. Advantages and disadvantages of such systems are discussed.

INTRODUCTION

Fuzzy logic was introduced in 1965 by Lotfi Zadeh [Ref. 1] as a method of capturing human expertise and incorporating this expertise into expert systems. The fuzzy set, which allows partial truths of statements as opposed to the classical Boolean logic, more correctly matches real world problems where people are forced into dealing with statements given in natural language. Statements of rules in natural language is a necessary way of dealing with decision making problems of the type typically handled by human experts. The use of the term "fuzzy logic" typically gives erroneous impressions to listeners that "fuzzy thinking" is involved. Actually, fuzzy logic is a mathematical method of dealing with situations that do not conform to Boolean logic. As an example, consider the following problem related to plant control. The condition of temperature being "low" is not a crisp concept, but is a condition that certain plant operators have to deal with on a regular basis. It is also not a probability concept as this operator is not the least bit interested in the probability that the temperature is low or even the question of whether it is likely that, given the current temperature, someone else might call it low. He is interested in evaluating the present situation, i.e., the present temperature that exists now, and deciding if it is low. This brings in the notion of a fuzzy set in a natural way since the condition of being low is a matter of degree. For example it seems to violate all rules of common sense to believe there is a particular temperature that satisfies the criterion that on the low side of that number, the temperature is low, while on the other side, it is not low. Mamdani and Assilian [Ref. 2] dealt with similar problems in their applications to ill-defined industrial processes.

Fuzzy set applications have been particularly successful in dealing with control of ill-defined processes of the type considered by Mamdani and Assilian. Excellent results for systems that are more well-defined but are still so complex that precise modeling is, at best, very difficult and expensive to do. Examples are the studies done at the Johnson Space Center (JSC) that use fuzzy sets in the control of space vehicle simulations and control of sensor data processing [Refs. 3,4]. The controller discussed in the following Control Applications section differs from the work in [Ref. 3] in the following respect. Although that controller is modeled on rules developed through conversations with simulator pilots, it was a hybrid type controller in that it used simple models to determine rates of the active vehicle and assumed that the measurements were smoothed and quite accurate when the controller did its evaluations to determine actions to take. This new variation of the controller uses rules modeled with fuzzy sets only for both rates and positions and uses the data from the sensors directly or with at most, very simplistic smoothing filters.

CONTROL APPLICATIONS

A controller that models the actions of a pilot as closely as is feasible has been modeled. Seven rules are used to control each of the velocities of the vehicle in its body, x, y, and z directions, respectively. As an example, let theta represent the elevation angle of the active vehicle with respect to the target vehicle, d_theta represent the corresponding angle rate, and let x_dot represent the velocity of the active vehicle in the x-direction. Let PS, PM, NS, NM, and ZO represent positive small, positive medium, negative small, negative medium, and zero, respectively. The set of rules are the following.

Rule 1. If theta is PM and d_theta is ZO, then x_dot is PM.

Rule 2. If theta is PS and d_theta is PS, then x_dot is PS.

Rule 3. If theta is PS and d_theta is NS, then x_dot is ZO.

Rule 4. If theta is NM and d_theta is ZO, then x_dot is NM.

Rule 5. If theta is NS and d_theta is NS, then x_dot is NS.

Rule 6. If theta is NS and d_theta is PS, then x_dot is ZO.

Rule 7. If theta is ZO and d_theta is ZO, then x_dot is ZO.

Also, rules for controlling the allowable rates need to be defined. These can be considered to be gain factors. These gain factors are important because allowable maximum rates will be smaller in the vicinity of the desired approach vector. Let A, L, VL, VVL denote average, large, very large, and very very large respectively for the gain factor. The additional fuzzy sets for theta, PL, NL, PVL, NVL, PVVL, and NVVL represent the conditions positive large, negative large, positive very large, negative very large, positive very very large and negative very very large respectively. The additional rules are specified below.

Rule 8. If theta is PS or PM or NS or NM, then gain is A.

Rule 9. If theta is PL or NL, then gain is L.

Rule 10. If theta is PVL or NVL, then gain is VL.

Rule 11. If theta is PVVL or NVVL, then gain is VVL.

Similar rules, as defined for elevation angle, are defined for the azimuth angle including gains since they typically will be different than for elevation control. Relative range rate, R_dot, control is specified by rules that make rate proportional to the relative range R, i.e., R_dot = k*R.

The value of k may be specified by a function of range, position, elevation angle, etc. However, for this study, k is a simple constant. The nominal value is .001 since this corresponds closely to rates maintained during manned rendezvous missions. It may be varied depending on the desired rate of approach and the relative range of the active and target vehicles. For R_dot control, rules are defined as a function of certain range gates. For example if r0, r1, r2, ... , rn are n-values of relative range and r0_dot, r1_dot, r2_dot, ..., rn_dot are n-values of relative range rate, then the range rate rules are defined as follows.

Rule 12. If R is less than r0, then R_dot should be approximately k*R.

Rule 13. If R is between r0 and r1, then R_dot should be about r0_dot.

Rule 14. If R is between r1 and r2, then R_dot should be about r1_dot.

.

.

.

Rule (12 + n). If R is between r(n-1) and rn, then R_dot should be about r(n-1)_dot.

Rule (13 + n). If R is greater than rn, then R_dot should be about rn_dot.

Typically, these values for range rate will be a function of the mission scenario, e.g., if the closing maneuver must be completed in a certain amount of time, then this will obviously affect the required closing rates. On the other hand, if the primary constraint is fuel usage then closing rates will be based on that consideration.

For this specific application development
r0 = 1000, r1 = 1500, r2 = 2000,
r3 = 4000, and r0_dot = 1.0,
r1_dot = 2.0, r2_dot = 3.0,
r3_dot = 4.0.
Therefore, there are 27 fuzzy rules for this simulation.

RESULTS

This version of the fuzzy controller has been compared to several other control sources. The results have been quite favorable although considerably more testing needs to be done. The current fuzzy pilot version more nearly conforms to rules that are adhered to by actual pilots of shuttle vehicles or simulators used for pre-mission analysis or training. The major problem that has not been completely solved as yet is the problem of noisy sensors where the noise is greater than the range in which the parameter under control needs to be maintained. This applies almost exclusively, for shuttle problems, to range rate control. Here the shuttle radar has a range rate 1-sigma noise of ±.3 ft/sec. One can easily understand the problems encountered when trying to control the system to .2 ft/sec when the noise is ±.3 ft/sec.

In comparisons with actual engineers and pilots flying the simulators using no additional aids than those available to the pilots, i.e., the modeling assumed no additional smoothing or massaging of the data, the fuzzy pilot performed perfectly acceptable so far as maintaining the desired trajectory and used within 5% of the propellant used by manned trajectory profiles. Some results from an early version of this controller [Ref. 5] support the above statement. With very unsophisticated smoothing of sensor data the automated controller will outperform the manual case. This is especially noticeable in stationkeeping maneuvers.

Further studies have been made to verify that fuzzy set models give results that are better than models using specifically crisp rules. Simulation tests for 40 cases, twenty which used crisp rules and twenty which used fuzzy rules, were made. Comparisons of the fuzzy and crisp cases were made based on propellant usage. In these runs the fuzzy rule-based controller performed, on the average, 20% better than the crisp rule-based controller. In cases where shuttle body rates were included in the simulation, which is the realistic case, the fuzzy controller performed better by 28%.

OTHER APPLICATIONS AND CONCLUSIONS

Rule-based controllers have been shown to be useful in control problems here at the JSC and also in numerous other studies and applications primarily in other countries. However, it should not be inferred that fuzzy controllers are always the best. When a process can be modeled quite accurately, and when this model can be used for estimating responses to actions in real-time there certainly seems to be merit for doing so. Bernard, and his associates at MIT, studied fuzzy rule-based versus analytic controllers in the control of a nuclear reactor power plant [Ref. 6] and came to the reasonable conclusion that each approach has its particular area of usefulness. For his particular application area he claims to get slightly better results with analytic controllers when data is close to the expected but that the fuzzy controller is more robust, i.e., it will tolerate much more unexpected situations and respond favorably. Thus it seems that fuzzy rule-based and analytic controllers may be best used together when modeling of the process is feasible. This is basically the philosophy taken in [Ref. 4] where fuzzy rules are used to control flow of data from various shuttle navigation sensors to an analytically derived Kalman filter, that does not perform well on data for which it has not been tuned.

REFERENCES

1. Zadeh, Lotfi, "Fuzzy Sets", Information and Control, vol. 8, 1965.

2. Mamdani, E. and S. Assilian, "An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller", Int. J. Man-Machine Studies, vol. 7, pp. 1-13, 1975.

3. Lea, R. N., "Automated Space Vehicle Control for Rendezvous Proximity Operations", Telematics and Informatics, vol. 5, No. 3, pp. 179-185, 1988.

4. Lea, R. N., "Applications of Fuzzy Sets to Data Monitoring and Process Control", Proc. of ISA International Conf., Houston, Tx., October, 1988.

5. Lea, R. N., Goodwin, M. A., and Mitchell, E. V., "Autonomous Control Procedures for Shuttle Rendezvous Proximity Operations", Proc. of the First Annual Workshop on Space Operations Automation and Robotics, pp. 281-286, August 1987.

6. Bernard, J. A., "Use of a Rule-Based System for Process Control", IEEE Control Systems Magazine, pp. 3-13, October 1988.

Late Submission

BLANK PAGE

Genetic Algorithms
Applied to the Scheduling of
The Hubble Space Telescope

Jeffrey L. Sponsler
Space Telescope Science Institute[*]
3700 San Martin, Baltimore, MD 21218 USA

Abstract

A prototype system employing a genetic algorithm (GA) has been developed to support the scheduling of the Hubble Space Telescope. A non-standard knowledge structure is used and appropriate genetic operators have been created. Several different crossover styles (random point selection, evolving points, and smart point selection) are tested and the best GA is compared with a neural network (NN) based optimizer. The smart crossover operator produces the best results and the GA system is able to evolve complete schedules using it. The GA is not as time-efficient as the NN system and the NN solutions tend to be better. Work is proposed to create a classifier system which can draw more effectively on the knowledge that is available in the scheduling domain.

1. Introduction

Genetic algorithms (Holland, 1975) are modeled from organic evolutionary systems and have been applied to search problems. Schaffer *et al* (1988) examined the evolution of crossover points (a technique used to minimize disruption of high performance schemata in chromosomes). Greffenstette (1988) has examined genetic algorithms with respect to rule discovery. De Jong (1980) applied GA technology to the problem of adaptive system design.

Genetic algorithm technology has been applied to real world problems such as scheduling. Hilliard *et al* (1988) have designed a competition-based system to discover scheduling heuristics. Baffes *et al* (1988) used greedy double-crossover techniques to generate optimal solutions to the space station mobile transporter scheduling problem.

This report describes on-going research wherein genetic algorithm technology has been applied to the problem of searching for feasible schedules for the Hubble Space Telescope and is organized as follows: First, a description of the Space Telescope and the constraint propagation system is given. Second, an informal analysis of the problem complexity is provided. Next, the genetic algorithm, crossover operators, and the results of experimentation that has been done are reported. Last, the results are discussed.

---

[*] Operated by the Association of Universities for Research in Astronomy for the National Aeronautics and Space Administration

2. The Hubble Space Telescope and SPIKE

The Hubble Space Telescope (HST) is an astronomical observatory that is to be placed into orbit in the near future by a space shuttle mission. The Space Telescope Science Institute is responsible for software ground support for HST. This involves, among other tasks, the processing of proposals (specifications for scientific experiments) submitted by astronomers in the international community. A proposal generally contains requests for activities (target acquisitions, exposures, and calibrations) and may include constraints upon these activities (e.g., "A before B"). The AI group at STScI has developed a system called **Spike,** which is a long-term scheduling utility. This system is described briefly below; for a more complete discussion of the Spike system, see Miller *et al* (1988).

2.1. Description of the Constraint-Based Scheduler

The Spike system supports and processes the following data objects: **targets** (stars, etc.), **activities** (exposures, etc.), **absolute constraints** (e.g., moon exclusion), and **relative constraints** (e.g., "A before B"). Each activity has an associated **suitability** that is a function of time and which provides knowledge about when it is legal to schedule an activity. A suitability is represented internally by a **piecewise constant function** (PCF) and is a list of time/value pairs (e.g., (minus-infinity 0 100 0.5 150 1 200 0.5 300 0)). Each absolute constraint is derived from appropriate astronomical models (e.g., the moon-exclusion constraint suitability is a function of target position and has value of 1 when the moon does not block the target and value of 0 otherwise). The suitability of a relative constraint is determined by looking at the activities that are linked via the constraint. For example, the suitability of the constraint "A before B" is calculated by looking at the legal times for A and doing the arithmetic to determine what times are legal for B such that B will follow A.

A group of activities linked via relative constraints is called a **dependency cluster**. Constraints may indirectly interact via common activity connections and so a **constraint propagation** technique is used. For example, if the explicit constraints "A before B" and "B before C" are specified, then "A before C" is an implicit constraint that is inferred via this propagation.

Spike is responsible for producing coarse year-long schedules consisting of time **segments** that are roughly week-long. Generating a finer schedule far in advance of execution is not reasonable (as the difference between present and future time increases the accuracy of HST orbit models decreases). The assignment of an activity to a time segment is called a **commitment**. This action causes the PCF of the activity to be zeroed over all time points outside of the selected time segment.

2.2. Complexity of Scheduling Problem

The scheduling of HST is an NP-complete problem. The generation and testing of all possible schedules is therefore practically impossible in cases where the numbers of things to be scheduled is large. In order to examine the complexity, let A be the number of activities to be scheduled and let S be the number of time segments in a schedule. The number of ways that one can assign A activities to S time segments is S^A.

The order in which the commitments in a possible schedule are made is important in judging the merits of a schedule. First, not all possible schedules can be instantiated; doing some subset of commitments may cause some other subset of commitments to be illegal. Second, in Spike the summed suitability of a schedule is used to rate the overall goodness; different orderings of the same set of activity/segment commitments will yield different results. Consider Figure 1 which illustrates how the ordering of commitments is important. On the basis of this, the complexity of the HST scheduling problem is $O(A!\ S^A)$.

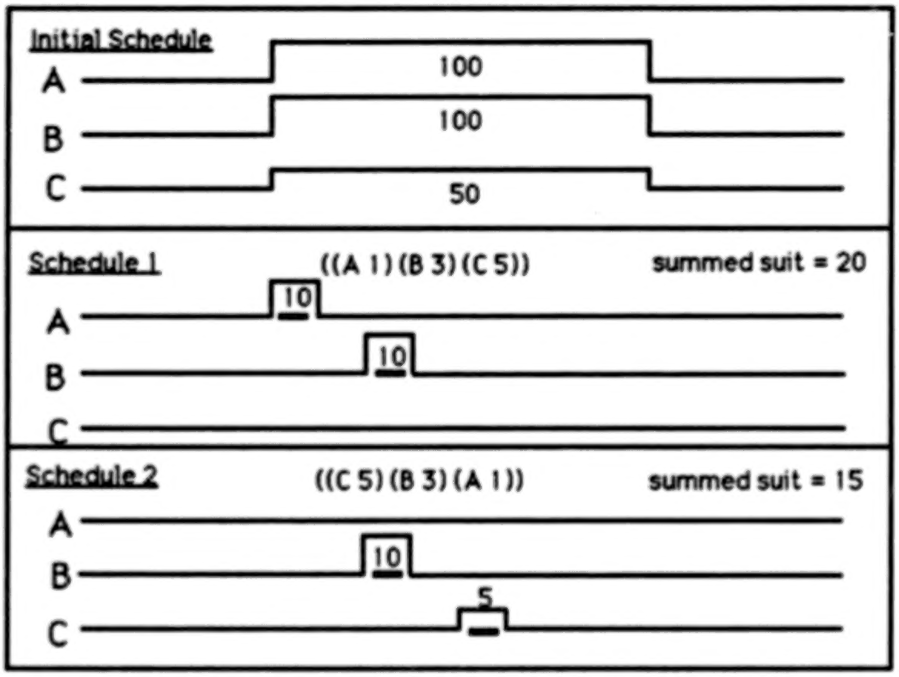


Figure 1. The ordering of commitments may yield different schedule suitabilities. In this contrived example, the relative constraint *within-2-time-units(A, B, C)* is implicit; this means that it is only legal to commit the activities so constrained such that no two are more than 2 time units apart. Committing activities A and B first in Schedule 1 disallows the (C 5) commitment (due to constraint violation) and produces suitability of 20; committing C and B first in Schedule 2 disallows the (A 1) commitment and produces suitability of 15. (A commitment is represented here as a small dark bar on a time line.)

2.3. Meta-Scheduling Techniques

The core Spike system has been implemented in a Common Lisp/Flavors/Common Windows environment. A user can manually create schedules or several procedural algorithms can be applied to automatically create schedules. The goal of the planning group is to use a meta-scheduling system to search for "optimal" solutions. A rule-based system and a neural network (NN) system (Johnston, 1989) have produced very promising results.

3. A Genetic Algorithm Applied to HST Scheduling

Genetic Algorithm technology has been applied to the HST scheduling problem in the form of a prototype system coded in Common Lisp that interacts with Spike and the neural network representation of commitments and constraints.

3.1. Representation of Knowledge Structures

The classic bit-string representation was not used in the GA prototype due to these factors: Crossover must yield complete schedules (one in which all activities are included) and the order of the commitments is important. Instead, the following representation was used. Each knowledge structure (**chromosome**) is a list consisting of sublists. Each sublist contains an activity id and a time segment number. An example chromosome illustrates: ((a 5) (b 7) (c 10) (d 10)). The ordering from left to right specifies the ordering of the commitment attempts. The interpretation of the example chromosomes is this: First attempt to commit activity a to time segment 5, then attempt to commit b to 7, and so on. All commitment pairs are attempted. The success of a given commitment attempt may have an affect on successive attempts. This is due to changes in suitabilities resulting from constraint propagation (i.e., commiting **a** to 5 may remove 10 as a valid time for **c**).

3.2. Crossover Techniques

In order to support exploration (via recombination) through the search space, two types of crossover are used. They have been called *horizontal* and *vertical.*

Vertical crossover is a binary operator and works in the following way. Let C_i and C_k be chromosomes. Select a site S_a in C_i where the desired cross is to occur. Then find the activity A_a at that site, and the corresponding site S_b in C_k where A_a resides. Swap the segments T_a at site S_a and T_b at site S_b. For example, if C_i is ((a 1) (b 2)(c 3)) and C_k is ((b 8) (a 7) (c 6)), swapping at site 2 in C_i will result in two new chromosomes: ((a 1) (b 8) (c 3)) and ((b 2) (a 7) (c 6)). This technique bears some resemblance to Goldberg's *partially matched crossover* (Goldberg, 1988). Vertical crossover can be applied iteratively over a range from one site to another.

Horizontal crossover is a unary operator and works as follows. Select two sites on the chromosome to be manipulated. Swap the two activity/segment pairs at those sites. For example, if the chromosome is ((a 1) (b 2) (c 3) (d 4)) and the sites are 2 and 4, the result will be ((a 1) (d 4) (c 3) (b 2)).

3.3. The Mutation Operator

The mutation operator works as follows: For a given chromosome iterate over each activity/segment pair. Flip a biased coin and if *true* results randomly select a new segment for the activity from the Table of Legal Segments.

3.4. The Genetic Algorithm

The genetic algorithm developed is not a general purpose parameter optimizer due to the unusual chromosome form. It does fit, however, into the general purpose design of the Spike scheduling system. Certain modifications to the algorithm form the basis for the experimentation reported here (details are found in later sections). Chromosomes are haploid. The pseudo-code algorithm and functional descriptions follow:

```
initialize-spike-system
create-table-of-legal-segments
setup-initial-population
process-chromosomes
loop-while (no-solution OR generations < max-gen)
   reproduce-chromosomes
   process-chromosomes
end-of-loop
```

To execute **initialize-spike-system,** proposals are selected from a pool and Spike data structures are instantiated as usual. In order to execute **create-table-of-legal-segments,** for each activity, a list of the legal segments where (at least initially) the activity could be committed is collected and stored in the Table with the activity.

The **setup-initial-generation** function creates a set of chromosomes of a specified size. To generate one chromosome, a complete set of randomly ordered activities is created. For each activity a segment number is selected randomly from the Table of Legal Segments.

The function **process-chromosomes** is responsible for interpreting (determining the fitness of) each chromosome (C_i) in a generation as well as calculating the average fitness of the generation and the relative fitness of each chromosome. The formula used to calculate a chromosome's fitness follows:

$$\text{fitness}(C_i) = \text{expt}(10 * \text{commitments/possible-commitments}) * \text{summed-suitability}(C_i)$$

The fitness function is heavily biased such that schedules with high commitment ratios are favored (the adjusted ratio is exponentiated with base of *e*). The motive for this was to highly reward the system for completed schedules. The function factors in the suitability of the schedule; this acts in a more subtle manner to differentiate schedules based on how good the commitments were.

Baker's Stochastic Universal Selection Process (Baker, 1988) is used to select offspring (surviving chromosomes) from the individuals in the generation. This technique selects offspring from a population based on the relative fitness of the individuals and is linear in complexity.

The step labeled **reproduce-chromosomes** executes crossover and mutation on each offspring. These actions are triggered with some specified probability.

3.5. Neural Network Used for Rapid Fitness Analysis

During chromosome interpretation, the Spike system is commanded to make a series of commitments. With each commitment, expensive processing is done (mostly to propagate relative constraints). In order to minimize this processing the following approach has been taken. A neural network is created that stores (as connection strengths) what changes occur to suitabilities when any given commitment is made. This network is then used to execute fitness evaluation. This approach has decreased GA processing time by at least one order of magnitude; the number of disposable *cons cells* generated is also much lower.

4. Description of Tested Hypotheses

4.1. A Smart Horizontal Crossover Technique

A modification on the basic random crossover technique has been developed because the random crossover was inefficient in achieving goal states. The *smart horizontal crossover* operator is knowledge-based and works as follows: Given two chromosomes C_i and C_k that have had fitness evaluations and have been selected as a mating pair, randomly select one chromosome (e.g., C_i). Consider each site (activity/segment pair) on C_i. If a site coded for a legal commitment (within the context of the ordered set of sites) then do nothing. If the site did not code for a legal commitment (as a result of previous commitments), execute a single vertical crossover with the appropriate site on the other chromosome (e.g., C_k). This technique will tend to preserve good regions regardless of length for one of the two chromosomes. Figure 2 illustrates.

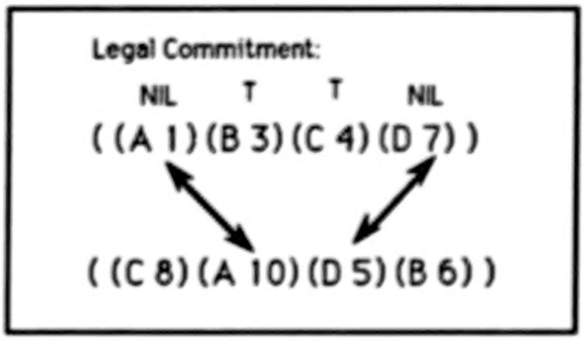


Figure 2. Smart horizontal crossover preserves high quality schemata. A T indicates a successful commitment and a NIL indicates an unsuccessful one.

The following hypotheses have been tested and the results will be presented in a later section.

H_0: A smart crossover will cause the GA to approach a solution state in roughly the same number of generations as that of a GA using a Random Crossover.

H_1: The GA employing Smart Crossover will reach a solution state in fewer generations than a GA using Random Crossover.

4.2. Evolution of Crossover Points

An algorithm that supports the evolution of crossover points has been implemented. Let C1 and C2 be the parent chromosomes. Associated with each are two crossover points. During crossover, one of the parents is randomly selected as the source of the two crossover points used for vertical crossover (which operates as usual). Let C3 and C4 be the offspring. C3 will inherit the crossover points from C1 and C4 will inherit from C2. Crossover points are subjected to mutation with frequency that is based on the specified probability of that operator. A crossover point mutation is merely the random selection of a new chromosome site number.

> H_0: An evolving crossover will cause the GA to approach a solution state in roughly the same number of generations as that of a GA using a random crossover.
>
> H_1: The GA employing the evolving crossover algorithm will reach a solution state in fewer generations than a GA using Random Crossover.

4.3. GA technology vs Neural Network

Since a neural network-based search algorithm is available, it seemed only reasonable to compare the two technologies. It was not expected that the GAs would perform as well as the NNs with respect to time. The summed suitability of solutions however was compared as a means to determine which technology produced more optimal solutions.

5. Experimental Results

Figures 3 and 4 illustrate the results from the experiments. The number of activities was 10 and each activity was constrained in several ways. From ten trials, an "average" run was selected from the Random and Smart results; the change in number of commitments over time is seen in Figure 3. Also from ten trials, the best run has been selected from Random, Evolving, and Smart Crossover trials; these results are seen in Figure 4.

In each trial, the probability of mutation was 0.03, the probability of crossover was 0.6, the population size was 30, and maximum allowed generations was 51.

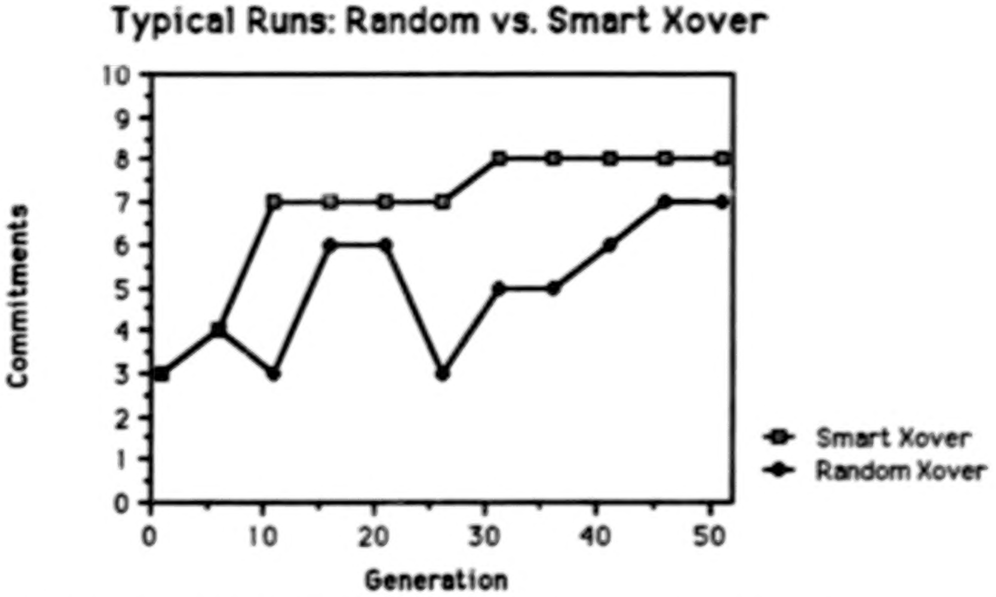


Figure 3. Smart vs Random Crossover. The data presented here were selected from "average" runs for both algorithms. The solution goal was 10 commitments.

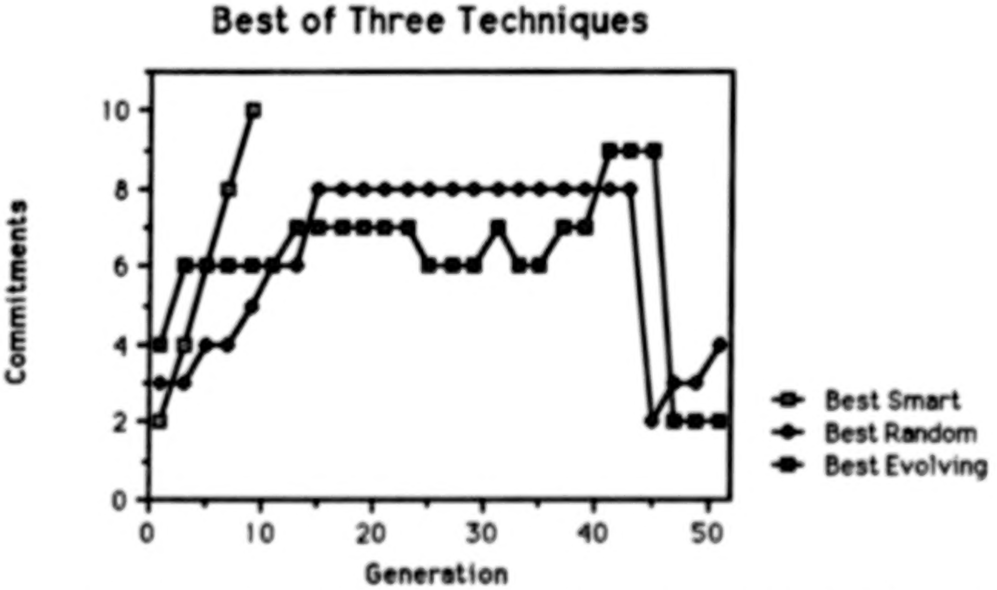


Figure 4. Random vs Evolving vs Smart Crossover. For each algorithm, the best (highest fitness) run has been plotted. The number of trials was 10 and the solution goal was 10 commitments.

5.1. Smart Crossover

Table 1 compares the performance of Smart versus Random Crossover. In all trials, the solution goal was 30 commitments, the population size was thirty, and the number of generations was thirty-one. The data indicate that the Smart Crossover produces statistically better results. The null hypothesis (see Section 4.1) is rejected and the true hypothesis is supported.

Xover	No. of Trials	Mean	Stand Dev	Z stat
Random	10	19.8	1.6	
Smart	10	25.1	1.79	6.98

Table 1. Random versus Smart Crossover. The z statistic has been calculated to determine whether the trial samples have been drawn from the same population and seem to indicate otherwise.

5.2. Crossover Point Evolution

The comparison of Random and Evolving Crossover Point algorithm is summarized in Table 2. The two populations are not significantly different and so the null hypothesis (see Section 4.2) regarding this technique is supported.

Xover	No. of Trials	Mean	Stand Dev	Z stat
Random	10	19.8	1.6	
Evolving	10	19.5	0.7	0.5

Table 2. Random versus Evolving Crossover. The z statistic has been calculated to determine whether the trial samples have been drawn from the same population and this is indicated.

5.3. GA vs NN

Tables 3 and 4 contain the results of comparisons of the two algorithms. The neural network algorithm, when applied to the same proposal as the Genetic Algorithms, arrived at solutions in much less time (generally a few seconds); the GA took many minutes to compute solutions (or near solutions). The solutions that were acheived by the NN tended to be better than the GA solutions.

Algorithm	N	Mean Secs to Soln	Stand Dev	Z stat
NN	5	4.8	0.8	
GA	5	1435.0	133.0	24

Table 3. Comparison of two search optimization algorithms with respect to speed. In all trials, >90% commitment of activities was achieved. The Z statistic indicates that the samples are not from the same population.

Algorithm	N	Mean Suitability	Stand Dev	Z stat
NN	5	18.0	0.1	
GA	5	9.9	1.3	13.9

Table 4. Comparison of GA and NN optimization techniques with respect to schedule suitability. The Z statistic indicates that the samples are not from the same population.

6. Discussion

The experimental results concerning the Smart Crossover algorithm indicate that this method tends to more quickly approach a solution; exploitation seems to be favored at the expense of exploration however and this may reduce the chance of finding a global optimum. The rapid movement by the GA to a solution may be a good thing in this scheduling domain; the computational expense of the GA (and the associated fitness testing) requires that population size and number of generations be kept to reasonable numbers.

The Random Cossover algorithm seems to more casually explore the search space. Looking at Figure 3, one sees empirical evidence for this in the oscillations and occasional radical backtracking to low fitness populations. The latter occurs after long periods of little change (and so may represent the probability that a higher than average number of mutations occurred in a single generation).

It is unclear why the Evolving Crossover Point performed no better than Random Crossover. Tools for tracing the evolution of single chromosomes have been proposed and could perhaps aid in "debugging" GA runs.

Continued work on this application of GA technology may yield better performing systems. There is a wealth of domain knowledge (constraint specifications) that can be tapped and so the design of a competition-based rule induction system should be a part of future work.

Acknowledgements

The author thanks the following persons for their ideas and comments concerning this work: Lashon Booker, Ken De Jong, John Greffenstette, Mark Johnston, Glenn Miller, and Shon Vick.

BEST COPY AVAILABLE

References

Baker, J. (1988). Reducing bias and inefficiency in the selection algorithm. *Proceedings of the Second International Conference on Genetic Algorithms* (pp. 14-21). Hillsdale, NJ: Lawrence Erlbaum Associates, Pub.

Baffes, P. and Wang, Lui (1988). Mobile transporter path planning using a genetic algorithm approach. *Proceedings of the SPIE Cambridge Symposium on Advances in Intelligent Robotics Systems*, vol. 1006 (pp. 226-234).

De Jong, K. (1980). Adaptive system design: a genetic approach. *IEEE Transactions on Systems, Man, and Cybernetics* , SMC-10(9) (pp. 566-574).

Grefenstette, J. (1988). Credit assignment in genetic learning systems. *Proceedings of the Seventh National Conference on Artificial Intelligence* (pp. 596-600). San Mateo, CA: Morgan Kaufman Publishers, Inc.

Goldberg, D. (1988). Genetic algorithms in search, optimization, and machine learning. Reading, MA: Addison-Wesley Pub. Co. Inc.

Hilliard, M., G. Liepins, and M. Palmer (1988). A classifier-based system for discovering scheduling heuristics. *Proceedings of the Second International Conference on Genetic Algorithms* (pp. 231-325). Hillsdale, NJ: Lawrence Erlbaum Associates, Pub.

Holland, J. (1975). *Adaptation In Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press.

Johnston, M. and H. Adorf (1989). Learning in stochastic neural networks for constraint satisfaction problems. *Proceedings of the NASA Conf. on Space Telerobotics* .

Miller, G., M. Johnston, S. Vick, J. Sponsler, and K. Lindenmayer (1988). Knowledge based tools for Hubble Space Telescope planning and scheduling: constraints and strategies. *Proceedings of the 1988 Goddard Conference On Space Applications of Artificial Intelligence.* Reprinted in *Telematics and Informatics* 5 (1988) (pp. 197-212).

Schaffer, J. and Morishima, A. (1988). Adaptive knowledge representation: a content sensitive recombination mechanism for genetic algorithms. *International Journal of Intelligent Systems*, Vol 3 (pp. 229-246).

National Aeronautics and Space Administration

Report Documentation Page

1. Report No. NASA CP-3033	2. Government Accession No.	3. Recipient's Catalog No.

4. Title and Subtitle 1989 Goddard Conference on Space Applications of Artificial Intelligence	5. Report Date April 1989
	6. Performing Organization Code 500
7. Author(s) James Rash, Editor	8. Performing Organization Report No. 89B00099
	10. Work Unit No.
9. Performing Organization Name and Address Goddard Space Flight Center Mission Operations & Data Systems Directorate Greenbelt, Maryland 20771	11. Contract or Grant No.
	13. Type of Report and Period Covered Conference Publication May 16-17, 1989
12. Sponsoring Agency Name and Address National Aeronautics & Space Administration Washington, D. C. 20546	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

This publication comprises the papers presented at the 1989 Goddard Conference on Space Applications of Artificial Intelligence held at the NASA/Goddard Space Flight Center, Greenbelt, Maryland, on May 16-17, 1989. The purpose of this annual conference is to provide a forum in which current research and development directed at space applications of artificial intelligence can be presented and discussed. The papers in these proceedings fall into the following areas: Mission Operations Support, Planning & Scheduling, Fault Isolation/Diagnosis, Image Processing & Machine Vision, Data Management, and Modeling & Simulation.

17. Key Words (Suggested by Author(s)) Artificial Intelligence, expert systems, mission operations support, planning and scheduling, fault isolation, fault diagnosis, image processing, machine vision, data management, modeling, simulation.	18. Distribution Statement Unclassified - unlimited Subject Category 63

19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of pages 412	22. Price A18

NASA FORM 1626 OCT 86

NASA-Langley, 1989

BEST COPY AVAILABLE

National Aeronautics and
Space Administration
Code NTT-4

Washington, D.C.
20546-0001

Official Business
Penalty for Private Use, $300

National Aeronautics and
Space Administration

Washington, D.C.
20546

SPECIAL FOURTH CLASS MAIL
BOOK

Postage and Fees Paid
National Aeronautics and
Space Administration
NASA-451

Official Business
Penalty for Private Use $300

POSTMASTER: If Undeliverable (Section 158 Postal Manual) Do Not Return

404

END

DATE FILMED

11-19-91